ESCUELA TÉNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
INGENIERÍA DEL SOFTWARE

IMPLEMENTACIÓN DE UNA APLICACIÓN WEB PARA LA REALIZACIÓN DE DIAGRAMAS DESTINADOS AL APRENDIZAJE BASADO EN PROBLEMAS DE ASIGNATURAS DE REDES

IMPLEMENTATION OF A DIAGRAMS CONSTRUCTION WEB APPLICATION FOR PROBLEM-BASED LEARNING APPROACH IN A SUBJECT ON COMPUTER NETWORKS

Realizado por
FABIÁN LUIS PLAZA ARNAU
Tutorizado por
MERCEDES AMOR PINILLA
EDUARDO GUZMÁN DE LOS RISCOS
Departamento:
LENGUAJES Y CIENCIAS DE LA COMPUTACIÓN

UNIVERSIDAD DE MÁLAGA
MÁLAGA, FEBRERO 2017

Fecha defensa:
El Secretario del Tribunal

Resumen: El objetivo principal de este trabajo es la realización de una aplicación web que permita al alumnado aprender, a través de la resolución de problemas, los principios de diseño, construcción y configuración de redes telemáticas permitiéndoles construir y configurar de forma adecuada diversas configuraciones de la misma, atendiendo a las características de la infraestructura subyacente.

Además, se pretende que, dicha aplicación web, sea flexible, dinámica y que permita al profesorado configurar sus propios componentes y las propiedades que lo definen, permitiendo a su vez que la aplicación siga creciendo y desarrollándose incluso acabado este proyecto.

Para ello, entre otros aspectos, y en el marco de este proyecto, se realizarán las siguientes actuaciones:

- Una aplicación web que permita el uso diferenciado de roles, disponiendo cada rol de las siguientes funciones:

Para el profesorado:

- Gestionar los componentes para los diagramas.
- Gestionar las propiedades que tendrán disponibles los componentes y que definirán, mediante expresiones regulares, la validez de éstas dentro del componente.
- Poder visualizar, calificar y comentar las tareas que el alumnado ha ido entregando.
- Posibilidad de configurar el acceso al alumnado en modo aprendizaje o modo examen.

Para el alumnado:

    - Poder visualizar todos aquellos diagramas creados, así como las observaciones y calificaciones que el profesorado ha realizado sobre la tarea en cuestión.
    - Poder realizar los diagramas de red y desde el que estén disponibles todos los componentes y propiedades que el profesorado haya registrado en los paneles destinados a ello.
    - Además desde este panel el alumnado podrá realizar tareas de:
        - generación de informes
        - exportar e importar JSON para guardar/visualizar diagramas ya creados.
        - Guardar el estado actual del lienzo creado
        - Entregar un lienzo creado para su corrección.
        - Simular envíos de paquetes entre diferentes componentes.

En conclusión, el conjunto de requisitos tiene como objetivo la creación de una aplicación web que permita a los alumnos practicar ejercicios relacionados con la construcción y configuración de diagramas de red.

Palabras clave: Diagramas de red, Ajax, PHP, jQuery, profesor, alumno.

Abstract: The main objective of this project is build an web application that allow to the students learn, through the resolution of problems, the basic principles of design, construction and configuration of telematic networks allowing them to construct and configure in an appropiate way multiples configurations of the same, attending the characteristics of the subjacent infrastructure.

Also, the intention is make an web application flexible, dynamic and that allow to the faculty to configure her own components and properties to define them, allowing that the application can grown when this project has finished.

For that, among other aspects, in the framework of this project, the following actions will be carried out:

- A web application that allow the differentiated use of roles, having each role the following functions:

For Teachers:

- Manage the components for diagrams.
- Manage the properties that will have the components and that will define, by regular expressions, the validity of them within of components.
- Can to visualize, calificate and comment the tasks that the students has entregate.
- Have the posibility to configure the students access in learning or exam mode.

For students:

- Can to visualize all created diagramas, as well the observations and scores of teachers.
- Can make the network diagrams and in which they have available all components and properties that the teacher has registered.
- Also, from this panel, the student could to make the next tasks:
    - Generate reports.
    - Export and import JSON to save/visualizate networks diagrams maked.
    - Save the actual state of networks diagrams.
    - To entregate the networks diagrams for her correction.
    - To simulate the packages's send through different components.

In conclusion, the final set of requirements have the objective of the creation of a web application that allow to the students make some exercises related with the build and configuration of networks diagrams.

Keywords: Network diagrams, Ajax, Php, JQuery, Teacher, student.

Índice:

Índice: 7
Tabla de ilustraciones: 9
1. Introducción 11
1.1 Objetivos 11
1.1.1 Tecnologías y herramientas utilizadas 12
1.2. Estructura de la memoria 14
1.2.1. Tecnologías utilizadas 14
1.2.2 Análisis de requisitos 14
1.2.3. Implementación y pruebas 14
1.2.4. Conclusiones 14
2. Tecnologías y herramientas utilizadas 15
2.1. PHP. 15
2.2. JQuery 16
2.3. AdminLTE, Bootstrap o Sass. 16
3. Análisis de requisitos. 19
3.1 Análisis 19
3.2 Requisitos 19
3.2.1. Requisitos no funcionales. 19
3.2.2. Requisitos funcionales. 20
3.2.3 Otros requisitos funcionales. 22
3.3. Actores 23
3.4 Diagramas de casos de uso 23
4. Diseño 25
4.1 Descripción de los casos de uso 25
4.1.1 Casos de uso asociados al profesor 25
4.1.2: Casos de uso asociados al alumno 32
4.2 Modelo de clases 38
4.2.1 Diagramas 38
5. Implementación 41
5.1 Primera iteración: BBDD y Clases PHP 42
5.1.1 Diseño del esquema de la BBDD 42
5.1.2 Diseño y creación de las clases PHP 44
5.2 Segunda Iteración: UI y Framework AdminLTE. 51
5.3 Tercera Iteración: UI Expresiones regulares y su Controlador. 52

5.3.1: Listado de expresiones regulares: 52
5.3.2: Proceso de añadir y Editar expresiones regulares 53
5.3.3: Validación de expresión regular 55
5.3.4: Controlador Expresiones_Regulares_Controller.php 56
5.4 Cuarta Iteración: UI Componentes y su Controlador 56
5.4.1: Listado de componentes 57
5.4.2: Proceso de añadir/Editar componentes 58
5.4.3: Controlador Elementos_Controller.php 59
5.5 Quinta Iteración: Panel de Ejercicios y Core.js 60
5.5.1: Importación dinámica de ficheros javascript 60
5.5.2: Clonación de Objetos al lienzo 62
5.5.3: Resto de funciones destacadas 63
5.6 Sexta Iteración: UI Corrección de lienzos. 65
5.6.1 Precargar lienzo 65
5.6.2 Comentar y calificar un lienzo 67
6: Conclusiones 69
6.1: Posibles mejoras 70
BIBLIOGRAFIA 71
ANEXO I: Manual del usuario 73
AN1: Navegación. 73
AN2: Expresiones Regulares 73
AN2.1: Añadir y editar una expresión regular 75
AN2.2: Validar una expresión regular (ER) 76
AN3: Gestor de componentes 77
AN3.1 Añadir y editar un componente 79
AN4: Lienzos 81
AN5: Panel de ejercicios 82
AN5.1: Propiedades. 85
AN5.2: Listado de conexiones y tarjetas de red 87
ANEXO II: Instalación 89

Tabla de ilustraciones:

Ilustración 1: Diagrama de casos de uso para el rol prrofesor. 23
Ilustración 2: Diagrama de casos de uso para el rol Alumno 24
Ilustración 3: Diagrama de clases Controladores..... 38
Ilustración 4: Diagrama de clases Modelos 38
Ilustración 5: Diagrama de clases (Otros)..... 38
Ilustración 6 - Tablas de iRedes 42
Ilustración 7 - Ejemplo de llamada AJAX..... 45
Ilustración 8 - Ejemplo de invocación a función add_action..... 46
Ilustración 9 - Implementación función add_action 46
Ilustración 10 - Implentación función do_action 47
Ilustración 11 - Métodos definidos en la Interfaz IController 48
Ilustración 12 - Métodos clase Model 50
Ilustración 13 - Menú lateral iRedes 51
Ilustración 14 - Solicitud de listado (Diagrama de Secuencias)..... 52
Ilustración 15 - Alta de una Expresión Regular (Diagrama de Secuencias) 53
Ilustración 16 - Edición de una Expresión Regular (Diagrama de Secuencias)..... 53
Ilustración 17 - Validación de una Expresión Regular (Diagrama de Secuencia)..... 55
Ilustración 18 - Estructura de la clase Expresiones_Regulares_Controller 56
Ilustración 19 - Solicitud de listado (Diagrama de Secuencias)..... 57
Ilustración 20 - Alta de un componente (Diagrama de Secuencia) 58
Ilustración 21 - Editar un componente (Diagrama de Secuencia) 58
Ilustración 22 - Métodos de Elementos_Controller 59
Ilustración 23 - Función cargar componentes..... 61
Ilustración 24 - Objeto JavaScript (Concentrador) 62
Ilustración 25 - Función precargar_lienzo..... 65
Ilustración 26 - Solicitud de lienzo para precarga (Diagrama de secuencia)..... 66
Ilustración 27 - Función comentar lienzo 67
Ilustración 28 - Comentar un lienzo (Diagrama de Secuencia) 67

1. Introducción

Este trabajo se enmarca en el Proyecto de Innovación Educativa de la Universidad de Málaga PIE13-115. Aprendizaje basado en Asignaturas de Redes y cuyo objetivo principal es la realización de una aplicación web que facilite al profesorado la explicación de los conceptos básicos relacionados con las redes de ordenadores y sistemas distribuidos al alumnado de manera didáctica.

1.1 Objetivos

Como objetivo principal de este trabajo nos encontraremos la realización de una aplicación web que reúna, entre otros, los siguientes requisitos:

- Aplicación Web con componentes dinámicos que puedan ser modificados, programados e integrados de manera sencilla en la aplicación permitiendo que ésta pueda seguir creciendo en un futuro.
- Creación de un panel de componentes que serán utilizados en la aplicación web para crear los diagramas de red.
- Creación de un panel de propiedades que podrán ser usados y validados en la aplicación web a través de los componentes
- Creación de un panel administrativo donde el profesorado pueda corregir, comentar y visualizar los trabajos realizados por el alumnado
- Creación de un panel (lienzo) donde el alumnado pueda realizar ejercicios de:
  - Creación de diagramas de red
  - Configuración de diagramas de red
  - Simulación del envío de paquetes a través de los diferentes componentes del diagrama
  - Validación de componentes y las propiedades que hayan sido definidos en éstos.
- Implementación que permita la comunicación mediante JSON con Siette.
- Diferenciación por ROLES de las funciones disponibles dentro de la aplicación, diferenciándose estos roles en PROFESOR y ALUMNO.
- Permitir que la aplicación diferencie entre modo examen y modo aprendizaje filtrando las funcionalidades de la misma.

1.1.1 Tecnologías y herramientas utilizadas

En la siguiente tabla se describen las principales herramientas, plataformas de programación y librerías empleadas en este proyecto.

Lenguajes de programación:

PHP 5.6.XX

Lenguaje de código abierto muy popular y especialmente adecuado para el desarrollo web de contenido dinámico.

JQuery

Librería JavaScript que permite la manipulación de los diferentes elementos DOM o realizar llamadas AJAX.

Sass

Sass es un metalenguaje de Hojas de estilo en cascada (css).

Software:

PhpStorm 2016.1

IDE de programación desarrollado por JetBrains es uno de los entornos de programación más completos de la actualidad y permite la edición de código no solo de su lenguaje sino de muchos otros.

MySQL WorkBench

Herramienta visual de diseño de bases de datos que integra desarrollo de software, administración de bases de datos, diseño de bases de datos, creación y mantenimiento para el sistema de bases de datos mysql. Sucesor de DBDesigner 5.

Cacoo

Herramienta para la creación de diagramas online gratuita que permite no solo la generación sino la exportación en diferentes formatos del diagrama realizado.

Bases de Datos:

MySql

Sistema de gestión de bases de datos relacionales desarrollado bajo licencia GPL/Licencia comercial por Oracle Corporation. Considerada como la base de datos Open Source más popular del mundo.

Frameworks:

Bootstrap

Frameworks de código abierto para diseño de sitios y aplicaciones web.

AdmiLTE

Framework para la creación del frontend y permitir que sea adaptable al dispositivo (responsive).

1.2. Estructura de la memoria

La memoria se ha estructurado de tal forma que tratará de cubrir las etapas seguidas en cualquier desarrollo de software, permitiendo al lector observar la evolución de manera iterativa del planteamiento expuesto en la introducción.

1.2.1. Tecnologías utilizadas

En esta sección se detallarán los motivos por los que se han seleccionado las diferentes tecnologías mencionadas anteriormente para el desarrollo de la aplicación web.

1.2.2 Análisis de requisitos

Durante esta fase se describirá con más detalle los requisitos obtenidos de las reuniones con los tutores del proyecto agrupándolos acorde a su función, así como los caso de uso detallando, para cada uno, los actores, escenarios y post/pre condiciones a cumplir. El objetivo será obtener una definición del sistema a realizar y del diseño de diagramas de clase.

1.2.3. Implementación y pruebas

En la última fase del proceso se explicará lo más destacado en cuanto a problemas encontrados y soluciones propuestas.

1.2.4. Conclusiones

Finalmente se explicará a modo de resumen lo aprendido durante el desarrollo del proceso.

2. Tecnologías y herramientas utilizadas

A continuación se explicará la elección y el uso de las tecnologías más destacadas utilizadas

2.1. PHP.

PHP es un lenguaje de programación de uso general de código del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico.

Fue uno de los primeros lenguajes de programación del lado del servidor que se podían incorporar directamente en el documento HTML en lugar de llamar a un archivo externo que procese los datos. El código es interpretado por un servidor web con un módulo de procesador PHP que genera la página web resultante.

PHP ha evolucionado mucho desde sus comienzos por lo que ahora incluye también una interfaz de línea de comandos que puede ser usada en aplicaciones gráficas independientes. Puede ser usado en la mayoría de los servidores web al igual que en casi todos los sistemas operativos y plataformas.

Además es un lenguaje fácil de aprender para el principiante pero que ofrece muchísimas características avanzadas para los programadores profesionales. Testigos de esta afirmación bien pudieran ser Wikipedia, Facebook o Wordpress, tres grandes compañías que dejan constancia de hasta dónde se puede llegar con PHP.

Antes de comenzar el proyecto, se estudió la tecnología que se debía utilizar para desarrollar la aplicación. Inicialmente se pensó en utilizar Java. Es el lenguaje utilizado durante todos los años de carrera, por lo que además de ser un lenguaje familiar, reunía los requisitos para poder desarrollar una aplicación de estas características, y, por este mismo motivo, es por lo que fue descartado. PHP es un lenguaje que no se estudia durante la carrera y por tanto suponía un “pequeño” reto.

Como uno de los requisitos para el desarrollo era la utilización de jQuery se decidió finalmente que el candidato perfecto era PHP. Bien se podía haber usado NodeJS (En el lado del Servidor) con AngularJS o incluso Angular2 (En el lado del cliente), pero la complejidad de estos lenguajes es mayor y otro de los requisitos era que fuese una aplicación de fácil mantenimiento y a cualquier persona con los conocimientos adquiridos durante la carrera le resultaría más rápido y sencillo aprender correctamente PHP y no NodeJS o Angular en cualquiera de sus versiones.

2.2. JQuery

jQuery es una biblioteca multiplataforma de JavaScript que permite simplificar la manera de interactuar con documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web.

Sobre la elección de esta tecnología no hay demasiado que decir: tan solo que fue un requisito para el desarrollo de la aplicación aunque bien es cierto que hoy día es difícil pensar en realizar una aplicación web sin utilizar jQuery, Angular, TypeScript o directamente JavaScript.

2.3. AdminLTE, Bootstrap o Sass.

Entre las tecnologías utilizadas para la capa de presentación he utilizado:

- Bootstrap
- Sass
- AdminLTE

Hoy día es muy importante cuidar no solo la funcionalidad sino también la estética de las aplicaciones. Los usuarios no buscan solo una aplicación funcional, también ha de ser una aplicación “bonita”, y todo ello debe realizarse en el menor tiempo posible. Además, las aplicaciones web deben poderse ejecutar en diferentes dispositivos como PC, móvil, tablet, etc. sin perder funcionalidad.

Es por esto y para no tener que desarrollar una aplicación para cada tipo de dispositivo que nacieron diferentes conjuntos de frameworks o lenguajes como pueden ser los tres presentados aquí.

Mientras que Bootstrap es un framework para el diseño de sitios web, AdminLTE es una aplicación base que utiliza este framework para crear una interfaz adaptativa (responsive) que se adapte a cada tipo de dispositivo mediante un conjunto de reglas CSS y apoyándose en jQuery para la aplicación de efectos sobre los diferentes elementos que podemos encontrar en el árbol DOM de nuestra aplicación.

Además, Sass, que es un lenguaje de hojas de estilo, permite al desarrollador crear una hoja de estilo más clara, fácil de mantener y extensible que si utilizásemos únicamente CSS. Sass permite la creación de clases, variables globales y locales consistentes en una serie de selectores y pseudo-selectores que agrupan las reglas que serán aplicadas en nuestra hoja de estilos. Una vez compilado el IDE creará el equivalente CSS que será el que interprete el navegador.

Aunque AdminLTE incluye muchísimos componentes con una estética agradable, otros muchos componentes que iba a necesitar la aplicación no estarían disponibles

y tendría que crearlos a mano. Para aplicarle estilos a estos componentes es para lo que se ha usado Sass.

3. Análisis de requisitos.

3.1 Análisis

Se desea desarrollar una aplicación web que permita al alumnado aprender, a través de la resolución de problemas, los principios de diseño, construcción y configuración de redes telemáticas permitiéndoles construir y configurar de forma adecuada diversas configuraciones de la misma, atendiendo a las características de la infraestructura subyacente.

Como no existe una aplicación previa con características similares, pasaremos a detallar directamente los requisitos, actores y diferentes casos de uso a desarrollar para llevar a cabo la aplicación.

seguri3.2 Requisitos

En este apartado listaremos los requisitos funcionales y no funcionales que determinarán el comportamiento de nuestra aplicación y definirán las acciones que los usuarios podrán realizar en ella.

Para facilitar la trazabilidad de los requisitos se dividirán las tablas por actores.

3.2.1. Requisitos no funcionales.

Estos requisitos especificarán criterios que pueden usarse para juzgar la operación de un sistema en lugar de sus comportamientos específicos que corresponderían a los requisitos funcionales. Es por tanto que estos requisitos describen características del funcionamiento y no información a guardar ni funciones a realizar.

Los requisitos no funcionales del sistema analizado son:

Categoria	ID	Descripción
Seguridad	RNF-001	La aplicación debe estar preparada para realizar la futura comunicación con SIETTE permitiendo la diferenciación entre profesorado y alumnado.
Usabilidad	RNF-002	Cualquier operación realizada por el usuario (Profesor o alumno) será informada por pantalla informando de si se ha producido algún error o si la operación se realizó con éxito.
Usabilidad	RNF-003	El sistema comprobará que los datos introducidos son correctos para evitar la

		introducción de información errónea en el sistema.
Usabilidad	RNF-004	El sistema dispondrá de dos modos de funcionamiento: Aprendizaje y Examen.
Usabilidad	RNF-005	El sistema mostrará al usuario pistas durante la creación y validación de una configuración del lienzo
Usabilidad	RNF-006	El sistema permitirá a los usuarios maximizar el espacio de la pantalla de trabajo durante la confección de un lienzo.
Usabilidad	RNF-007	El sistema permitirá la utilización de atajos de teclas para permitir la facilidad de uso en cuanto a mover, enlazar/desenlazar o eliminar componentes se refiere.
Portabilidad	RNF-008	El sistema debe ser portable de manera que podamos instalar nuestra aplicación en cualquier tipo de servidor.
Costo	RNF-009	El coste de mantenimiento debe ser mínimo
Escalabilidad	RNF-010	El sistema debe ser fácilmente escalable pudiéndose añadir fácilmente nuevos componentes sin que eso afecte al sistema.
Mantenibilidad	RNF-011	El sistema debe ser fácilmente mantenible.
Interfaz	RNF-012	El sistema dispondrá de una interfaz clara, adaptativa (responsive) y diferenciada por roles.

3.2.2. Requisitos funcionales.

En este apartado se detallarán todos aquellos requisitos que describan una función a realizar por los diferentes roles de la aplicación. Algunas funcionalidades extra se detallarán en el apartado 3.2.3 de este documento.

3.2.2.1 Requisitos funcionales del Profesorado:

ROL	ID	Nombre	Descripción
Profesor	RF-P001	Registrar expresión regular	El profesor podrá dar de alta en el sistema expresiones regulares que definan la validez de las propiedades que conformarán un diagrama
Profesor	RF-P002	Registrar componentes	El profesor podrá dar de alta en el sistema componentes que podrán ser usados en el lienzo por los alumnos

Profesor	RF-P003	Calificar lienzo	El profesor podrá calificar los lienzos de los alumnos
Profesor	RF-P004	Comentar lienzo	El profesor podrá realizar comentarios sobre los lienzos de los alumnos
Profesor	RF-P005	Cargar lienzo	El profesor podrá realizar la carga de los lienzos realizados por los alumnos de manera rápida y sencilla.
Profesor	RF-P006	Editar registro	El profesor podrá realizar una edición de cualquier registro relacionado con las expresiones regulares, componentes o lienzos.
Profesor	RF-P007	Elimina registro	El profesor podrá eliminar de manera segura cualquier registro relacionado con las expresiones regulares y/o componentes.
Profesor	RF-P008	Testear expresión regular	El profesor podrá testear cualquier expresión regular registrada en el sistema.
Profesor	RF-P009	Consultar código JS asociado	El profesor podrá realizar la lectura del código asociado a un componente de manera sencilla sin tener que acceder al servidor para descargar le fichero.
Profesor	RF-P010	Consultar componentes	El profesor podrá consultar el listado de componentes registrados
Profesor	RF-P011	Consultar Expresiones Regulares	El profesor podrá consultar el listado de expresiones regulares registrados.
Profesor	RF-P012	Consultar lienzos	El profesor podrá consultar el listado de lienzos disponibles tanto propios como de los usuarios que hayan pulsado en entregar.

3.2.2.2. Requisitos funcionales del Alumnado:

ROL	ID	Nombre	Descripción
Alumno	RF-A001	Crear lienzo	El alumno podrá crear lienzos utilizando los componentes y propiedades que el profesor haya registrado en el sistema.
Alumno	RF-A002	Consultar lienzo	El alumno podrá consultar las calificaciones y comentarios de los lienzos que ha realizado.

Alumno	RF-A003	Cargar lienzo	El alumno podrá realizar la carga de lienzos previamente guardados.
Alumno	RF-A004	Generar informe	El usuario podrá realizar una validación de todos los componentes y propiedades de su configuración y disponer de un informe detallado de ésta.
Alumno	RF-A005	Exportar lienzo	El alumno debe poder realizar la exportación del JSON que contiene la configuración realizada en el lienzo.
Alumno	RF-A006	Importar lienzo	El alumno debe poder realizar la importación de un JSON que contenga una configuración válida del lienzo.
Alumno	RF-A007	Guardar estado lienzo	El alumno debe poder guardar el estado actual de la configuración de su lienzo.
Alumno	RF-A008	Entregar lienzo	El alumno debe poder marcar como entregado una configuración para que el profesor pueda realizar su corrección.
Alumno	RF-A009	Enviar paquete	El alumno debe poder realizar una simulación de envío de paquetes entre diferentes componentes de su configuración.
Alumno	RF-A010	Añadir componente	El alumno debe poder añadir componentes al lienzo
Alumno	RF-A011	Añadir propiedad	El alumno debe poder añadir propiedades a un componente
Alumno	RF-A012	Validar propiedad	El alumno debe poder validar las propiedades asociadas al componente
Alumno	RF-A013	Eliminación de un componente	El alumno debe poder eliminar de su configuración cualquier componente añadido.

3.2.3 Otros requisitos funcionales.

Otros requisitos funcionales de menor importancia pero que añaden funcionalidad a la aplicación son:

- RF-A014 – Actualizar propiedad: Permite actualizar el valor de una propiedad ya asignada a un componente.
- RFA015 – Eliminar propiedad: Permite eliminar una propiedad que ha sido asignada a un componente.

- RFA016 – Visualización propiedad: Permite visualizar en el lienzo una propiedad asignada a un componente para permitir tener una referencia visual de ésta en el lienzo.

3.3. Actores

A continuación se detallan los actores del sistema:

- **Sistema**: Representa la aplicación iRedes.
- **Profesorado**: Rol de la aplicación. Representa a aquella persona que imparta la clase y realizará las veces de administrador del sistema.
- **Alumnado**: Rol de la aplicación. Representa a todos aquellos alumnos inscritos en la materia y que realizarán las configuraciones en el lienzo.

3.4 Diagramas de casos de uso

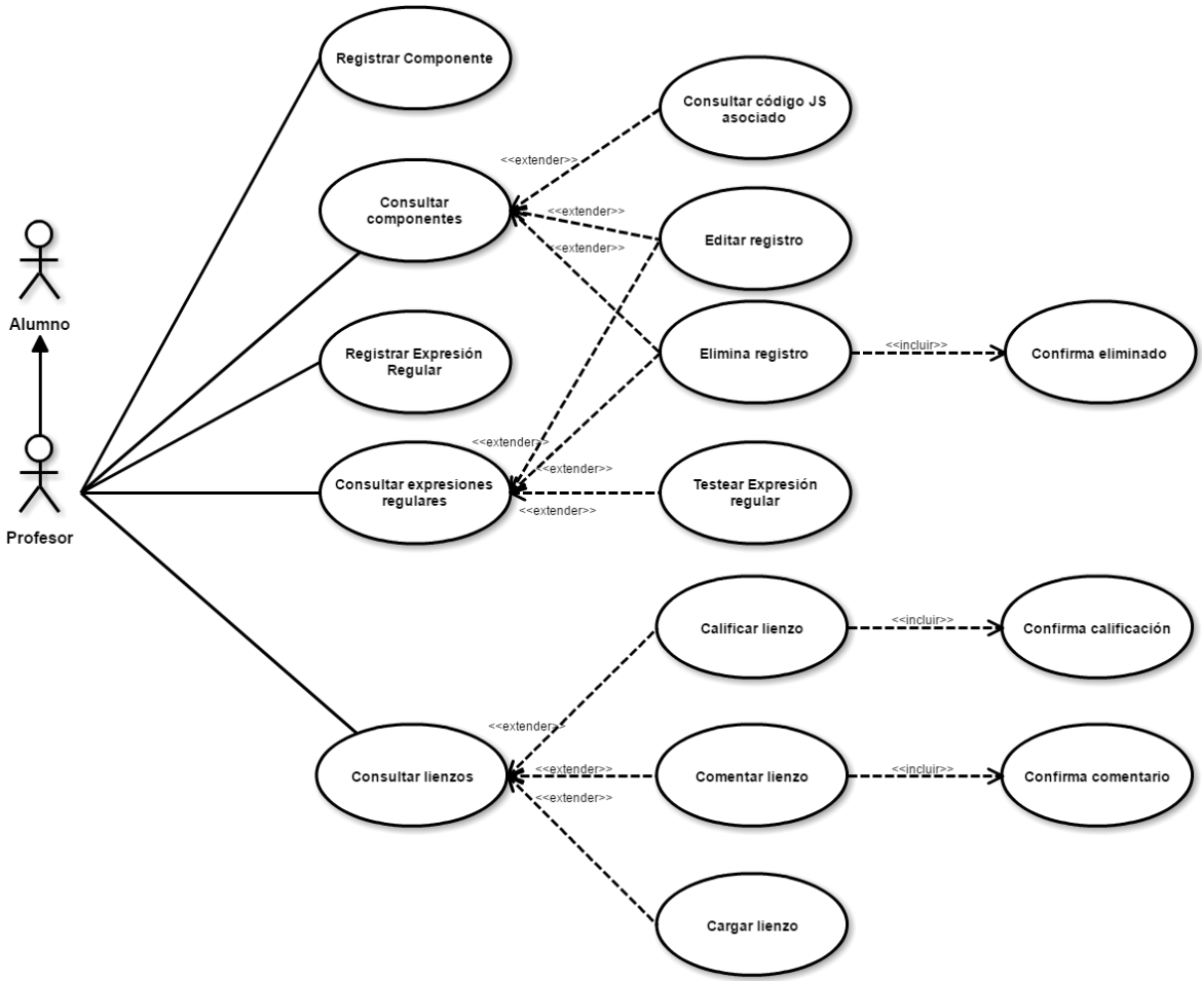

Ilustración 1: Diagrama de casos de uso para el rol profesor.

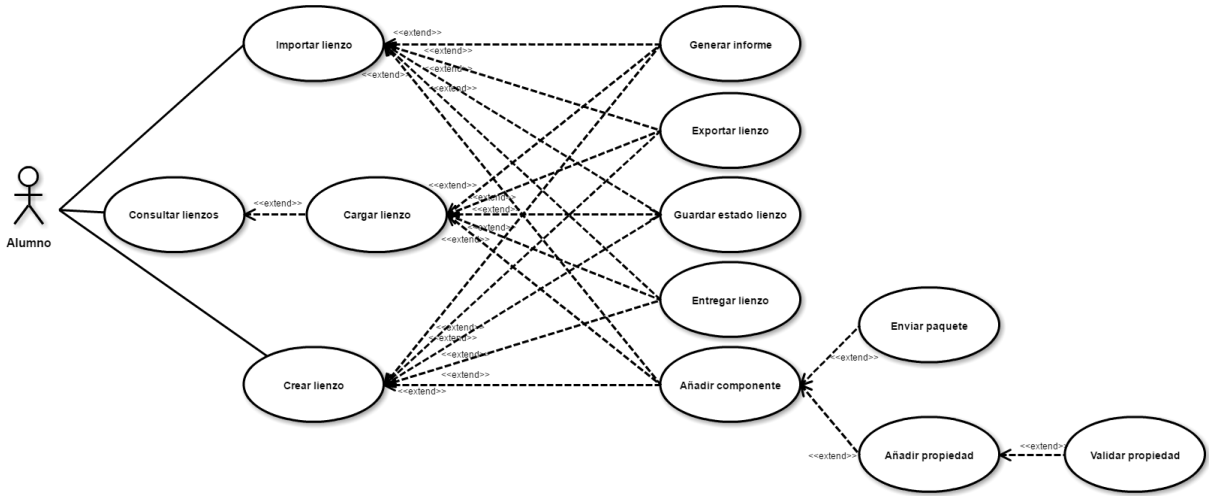

Ilustración 2: Diagrama de casos de uso para el rol Alumno

4. Diseño

4.1 Descripción de los casos de uso

A continuación se detallarán los casos de uso que recogen los principales escenarios de nuestra aplicación.

4.1.1 Casos de uso asociados al profesor

ID	**U1**
Caso de uso	Registrar Expresión regular
Actores	Profesor
Descripción	Registro de una expresión regular que posteriormente podrá usar el alumno para asociar a un componente que introduzca en su configuración del lienzo.
Precondiciones	El usuario debe estar autenticado como profesor
Escenario principal	1. El usuario hace clic sobre la sección expresiones regulares. 2. El sistema muestra una ventana que contiene el listado de todos los componentes registrados. 3. El usuario hace clic en el botón “Añadir”. 4. El sistema muestra una ventana con un formulario para la creación de expresiones regulares. 5. El usuario rellena correctamente los campos mostrados y pulsa el botón “Añadir”. 6. El sistema informa al usuario de que el alta se ha realizado correctamente
Escenario alternativo	7. El sistema informa al usuario del error ocurrido al intentar registrar la expresión regular.
Requisito Asociado	RF- P001

ID	**U2**
Caso de uso	Registrar componente
Actores	Profesor
Descripción	Registro de un componente que posteriormente puede ser utilizado en la realización de los diagramas.
Precondiciones	El usuario debe estar autenticado como profesor
Escenario principal	1. El usuario hace clic sobre la sección Gestor componentes. 2. El sistema muestra una ventana que contiene el listado de todos los componentes registrados. 3. El usuario hace clic en el botón “Añadir”.

	4. El sistema muestra una ventana con un formulario para la creación de componentes. 5. El usuario rellena correctamente los campos mostrados y pulsa el botón “Añadir”. 6. El sistema informa al usuario de que el alta se ha realizado correctamente
Escenario alternativo	7. El sistema informa al usuario de un error en el alta del componente
Requisito asociado	RF-P002

ID	**U3**
Caso de uso	Calificar lienzo
Actores	Profesor
Descripción	Calificación de un lienzo que un alumno ha realizado.
Precondiciones	El usuario debe estar autenticado como profesor
Escenario principal	1. El usuario pulsa sobre “Lienzos” en el menú lateral. 2. El sistema muestra el listado de lienzos de los que es propietario y los que los alumnos han marcado como “Entregados”. 3. El profesor pulsa sobre el botón rojo con el icono que indica la calificación del lienzo. 4. El sistema muestra ventana con un cuadro de texto donde el profesor podrá poner la calificación obtenida. 5. El profesor introduce la calificación usando el punto como separador decimal. 6. El sistema muestra un mensaje indicando que la calificación se ha realizado con éxito 7. El sistema actualiza el listado de lienzos con la nueva calificación.
Escenario alternativo	8. El sistema informa al usuario de un error en la calificación del lienzo.
Requisito asociado	RF-P003

ID	U4
Caso de uso	Comentar lienzo
Actores	Profesor
Descripción	Comentario sobre un lienzo que ha realizado un alumno
Precondiciones	El usuario debe estar autenticado como profesor
Escenario principal	1. El usuario pulsa sobre “Lienzos” en el menú lateral. 2. El sistema muestra el listado de lienzos de los que es propietario y los que los alumnos han marcado como “Entregados”. 3. El profesor pulsa sobre el botón azul con el icono que indica la acción de comentar lienzo. 4. El sistema muestra ventana con un cuadro de texto donde el profesor podrá poner un comentario al lienzo. 5. El profesor introduce el comentario deseado 6. El sistema muestra un mensaje indicando que el comentario se ha guardado con éxito.
Escenario alternativo	7. El sistema informa al usuario de un error durante la operación de comentar el lienzo.
Requisito asociado	RF-P004

ID	U5
Caso de uso	Cargar lienzo
Actores	Profesor, Alumno
Descripción	Carga de un lienzo guardado en el panel de ejercicios.
Precondiciones	El usuario debe estar autenticado como profesor o alumno
Escenario principal	1. El usuario pulsa sobre “Lienzos” en el menú lateral. 2. El sistema muestra el listado de lienzos de los que es propietario y/o los que los alumnos han marcado como “Entregados” en función del rol del usuario. 3. El usuario pulsa sobre el botón gris con el icono que indica la acción de cargar lienzo (Play).

	4. El sistema realizará una navegación a la sección Panel de Ejercicios y cargará el JSON asociado.
Escenario alternativo	7. El sistema informa al usuario de un error durante la operación de cargar JSON asociado.
Requisito asociado	RF-P005, RF-A003

ID	**U6**
Caso de uso	Editar registro
Actores	Profesor
Descripción	Edita un registro de un componente o una expresión regular.
Precondiciones	El usuario debe estar autenticado como profesor
Escenario principal	1. El usuario pulsa sobre “Expresiones Regulares” o “Gestor de componentes”. 2. El sistema muestra el listado de Expresiones regulares o componentes en función de la sección pulsada en el paso 1. 3. El usuario pulsa sobre el botón con el icono del lápiz para editar el registro seleccionado. 4. El sistema mostrará una ventana con la información del registro correspondiente. 5. El usuario modificará todos los datos que necesite del registro en cuestión y pulsará el botón guardar. 6. El sistema mostrará una alerta indicando al usuario que la operación se realizó correctamente.
Escenario alternativo	7. El sistema informa al usuario de un error durante la operación de cargar JSON asociado.
Requisito asociado	RF-P006

ID	**U7**

Caso de uso	Elimina registro
Actores	Profesor
Descripción	Elimina un registro de un componente o una expresión regular.
Precondiciones	El usuario debe estar autenticado como profesor
Escenario principal	1. El usuario pulsa sobre “Expresiones Regulares” o “Gestor de componentes”. 2. El sistema muestra el listado de Expresiones regulares o componentes en función de la sección pulsada en el paso 1. 3. El usuario pulsa sobre el botón con el icono de la basura para eliminar el registro seleccionado. 4. El sistema mostrará una ventana de confirmación para el borrado. 5. El usuario aceptará o cancelará la operación. 6. El sistema mostrará una alerta indicando que la operación se ha realizado o cancelado con éxito.
Escenario alternativo	7. El sistema informa al usuario de un error durante la operación de eliminación
Requisito asociado	RF-P007

ID	**U8**
Caso de uso	Testear expresión regular
Actores	Profesor
Descripción	Se realiza una comprobación de la expresión regular registrada con el fin de si pasará el test de validación o no una vez el usuario la utilice en el lienzo.
Precondiciones	El usuario debe estar autenticado como profesor
Escenario principal	1. El usuario pulsa sobre “Expresiones Regulares” . 2. El sistema muestra el listado de Expresiones. 3. El usuario pulsa sobre el botón verde con el icono de “check”

	4. El sistema mostrará una ventana donde el usuario podrá introducir el valor que desea testear para la expresión regular seleccionada. También podrá realizar (pero no guardar) modificaciones sobre la expresión regular con el fin de afinarla. 5. El usuario pulsará el botón “Comprobar” 6. El sistema cambiará el fondo de pantalla en función de si la expresión regular se ha satisfecho para el valor indicado (verde) o no (rojo). Además mostrará las coincidencias del valor para la expresión regular.
Escenario alternativo	No aplica.
Requisito asociado	RF-P008

ID	**U9**
Caso de uso	Consultar código JS asociado
Actores	Profesor
Descripción	Se realiza una comprobación de la expresión regular registrada con el fin de si pasará el test de validación o no una vez el usuario la utilice en el lienzo.
Precondiciones	El usuario debe estar autenticado como profesor
Escenario principal	1. El usuario pulsa sobre “Gestor componentes”. 2. El sistema muestra el listado de componentes registrados. 3. En el listado, el usuario pulsa sobre el enlace ubicado en la columna “JS”. 4. El sistema mostrará una ventana con el contenido del fichero para su consulta.
Escenario alternativo	No aplica.
Requisito asociado	RF-P009

ID	U10
Caso de uso	Consultar componentes
Actores	Profesor
Descripción	Se realiza una comprobación de la expresión regular registrada con el fin de si pasará el test de validación o no una vez el usuario la utilice en el lienzo.
Precondiciones	El usuario debe estar autenticado como profesor
Escenario principal	1. El usuario pulsa sobre “Gestor componentes”. 2. El sistema muestra el listado de componentes registrados.
Escenario alternativo	No aplica.
Requisito asociado	RF-P010

ID	U11
Caso de uso	Consultar expresiones regulares
Actores	Profesor
Descripción	Se realiza una comprobación de la expresión regular registrada con el fin de si pasará el test de validación o no una vez el usuario la utilice en el lienzo.
Precondiciones	El usuario debe estar autenticado como profesor
Escenario principal	1. El usuario pulsa sobre “Expresiones regulares” en el menú lateral. 2. El sistema muestra el listado de componentes registrados.
Escenario alternativo	No aplica.
Requisito asociado	RF-P011

4.1.2: Casos de uso asociados al alumno

(Nota: Todos los casos de uso descritos a continuación pueden ser también realizados por un Profesor)

ID	**U12**
Caso de uso	Creación de lienzos
Actores	Profesor, Alumno
Descripción	Permite la creación de una configuración de red en un lienzo.
Precondiciones	El usuario debe estar autenticado como profesor o alumno
Escenario principal	1. El usuario pulsa sobre Panel de ejercicios 2. El sistema muestra panel de ejercicios donde el usuario dispondrá de todas las opciones necesarias para la creación de la configuración.
Escenario alternativo	No aplica.
Requisito asociado	RF-A001

ID	**U13**
Caso de uso	Consultar lienzo
Actores	Profesor, Alumno
Descripción	Permite visualizar el listado de lienzos registrados. Dependiendo del rol mostraremos todos (profesor), entregados (profesor) o solo los propios (alumno)
Precondiciones	El usuario debe estar autenticado como profesor o alumno
Escenario principal	1. El usuario pulsa sobre la sección lienzos. 2. El sistema muestra el listado de lienzos disponible en base a la descripción dada para este caso de uso.
Escenario alternativo	No aplica.
Requisito asociado	RF-A001, RF-P012

ID	U14
Caso de uso	Generar informe
Actores	Alumno
Descripción	Permite al usuario generar un informe con la revisión y validez de todos los componentes que conforman la configuración del lienzo.
Precondiciones	El usuario debe estar autenticado como alumno
Escenario principal	1. El usuario pulsa sobre la sección Panel de ejercicios. 2. El sistema carga el panel de ejercicios. 3. El usuario realiza una de las siguientes acciones: 3.1. Importa el contenido de un JSON. 3.2. Crea componentes nuevos usando la botonera de componentes. 4. El usuario pulsa sobre el botón “Informe” de la botonera de acciones. 5. El sistema muestra por pantalla el informe global de los componentes.
Escenario alternativo	No aplica.
Requisito asociado	RF-A004

ID	U15
Caso de uso	Exportar lienzo
Actores	Alumno
Descripción	Permite al usuario exportar un lienzo para obtener el JSON
Precondiciones	El usuario debe estar autenticado como alumno Haber realizado los pasos descritos en el **U12**
Escenario principal	1. El usuario pulsa el botón “Exporta” de la barra de acciones.
Escenario alternativo	No aplica.
Requisito asociado	RF-A005

ID	U16
Caso de uso	Importar lienzo
Actores	Alumno
Descripción	Permite al usuario importar un lienzo para obtener el JSON
Precondiciones	El usuario debe estar autenticado como alumno Haber realizado los pasos descritos en el **U12**
Escenario principal	1. El usuario pulsa el botón “Importa” de la barra de acciones y pega el contenido del fichero JSON en el cuadro de texto. Pulsa finalmente el botón “Asignar”. 2. El sistema carga el contenido del fichero JSON.
Escenario alternativo	No aplica.
Requisito asociado	RF-A006

ID	U17
Caso de uso	Guardar estado lienzo
Actores	Alumno
Descripción	Permite al usuario guardar el estado de un lienzo.
Precondiciones	El usuario debe estar autenticado como alumno y haber realizado los pasos descritos en el **U12**
Escenario principal	1. El usuario pulsa el botón “Guarda” de la botonera de acciones. 2. El sistema muestra aviso de guardado correcto.
Escenario alternativo	3. El sistema muestra aviso de error en el guardado.
Requisito asociado	RF-A007

ID	U18
Caso de uso	Entregar lienzo
Actores	Alumno
Descripción	Permite al usuario entregar un lienzo para su corrección.
Precondiciones	El usuario debe estar autenticado como alumno Haber realizado los pasos descritos en el **U12**
Escenario principal	1. El usuario pulsa el botón “Entrega” de la botonera de acciones. 2. El sistema muestra aviso de entrega correcta.
Escenario alternativo	3. El sistema muestra aviso de error en la entrega.
Requisito asociado	RF-A008

ID	U19
Caso de uso	Enviar paquete
Actores	Alumno
Descripción	Permite al usuario entregar un lienzo para su corrección.
Precondiciones	El usuario debe estar autenticado como alumno Haber realizado los pasos descritos en el **U12** Haber introducido al menos 2 componentes en el lienzo.
Escenario principal	1. El usuario selecciona el componente de origen y destino. 2. El sistema los marcará en morado para indicar que han sido seleccionados. 3. El usuario pulsará el botón “envío paquete” 4. El sistema mostrará el número de saltos necesarios para realizar el envío del paquete.
Escenario alternativo	5. El sistema mostrará un error en caso de que la configuración de los componentes no sean válidos (No tengan tarjetas de red asociadas y lo necesiten, no tengan configurada una IP, etc.)
Requisito asociado	RF-A009

ID	U20
Caso de uso	Añadir componente
Actores	Alumno
Descripción	Permite al usuario añadir un componente al lienzo
Precondiciones	El usuario debe estar autenticado como alumno Haber realizado los pasos descritos en el **U12**
Escenario principal	1. El usuario pulsa sobre cualquiera de los iconos que representan los componentes en la barra de componentes (lateral izquierdo) 2. El sistema introduce el componente seleccionado en el lienzo para su configuración
Escenario alternativo	No aplica
Requisito asociado	RF-A010

ID	U21
Caso de uso	Añadir propiedad
Actores	Alumno
Descripción	Permite al usuario añadir un componente al lienzo
Precondiciones	El usuario debe estar autenticado como alumno Haber realizado los pasos descritos en el **U12 y U20**
Escenario principal	1. El usuario hace doble clic sobre el componente introducido. 2.El sistema muestra la ventana de propiedades del componente seleccionado. 3. El usuario introduce una propiedad en el componente. Esta propiedad puede ser una de las expresiones regulares registradas, una línea en la tabla de encaminamiento, una tarjeta de red o un comentario sobre el componente.
Escenario alternativo	No aplica
Requisito asociado	RF-A011

ID	U22
Caso de uso	Validar propiedad
Actores	Alumno
Descripción	Permite al usuario añadir un componente al lienzo
Precondiciones	El usuario debe estar autenticado como alumno Haber realizado los pasos descritos en el **U12, U20 y U21**
Escenario principal	1. El usuario pulsa el botón amarillo con el icono del “check” para realizar la validación de la propiedad. 2. El sistema muestra una alerta con la información sobre la validación de la propiedad.
Escenario alternativo	No aplica
Requisito asociado	RF-A012

ID	U23
Caso de uso	Eliminar componente
Actores	Alumno
Descripción	Permite al usuario eliminar un componente del lienzo
Precondiciones	El usuario debe estar autenticado como alumno Haber realizado los pasos descritos en el **U12 y U20**
Escenario principal	1. El usuario pulsa el botón **supr** después de haber seleccionado un componente **o** 2. El usuario hace **doble clic** sobre el componente y pulsa en la pestaña eliminar componente 3. El sistema solicita confirmación de borrado 4. El usuario pulsa aceptar 5. El sistema elimina el componente seleccionado
Escenario alternativo	6. El usuario cancela la eliminación del componente 7. El sistema avisa de la cancelación
Requisito asociado	RF-A012

4.2 Modelo de clases

4.2.1 Diagramas

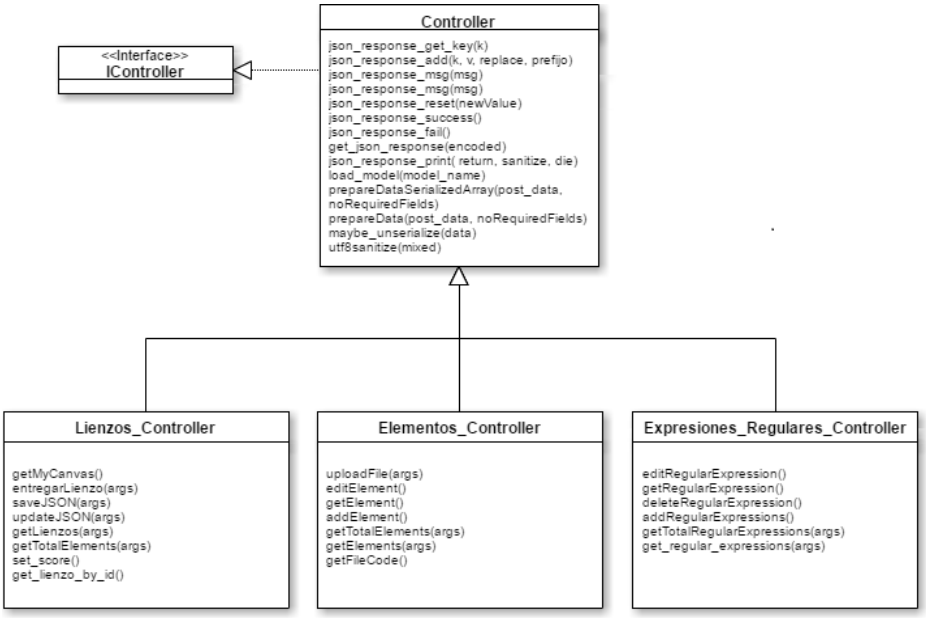

Ilustración 3: Diagrama de clases Controladores

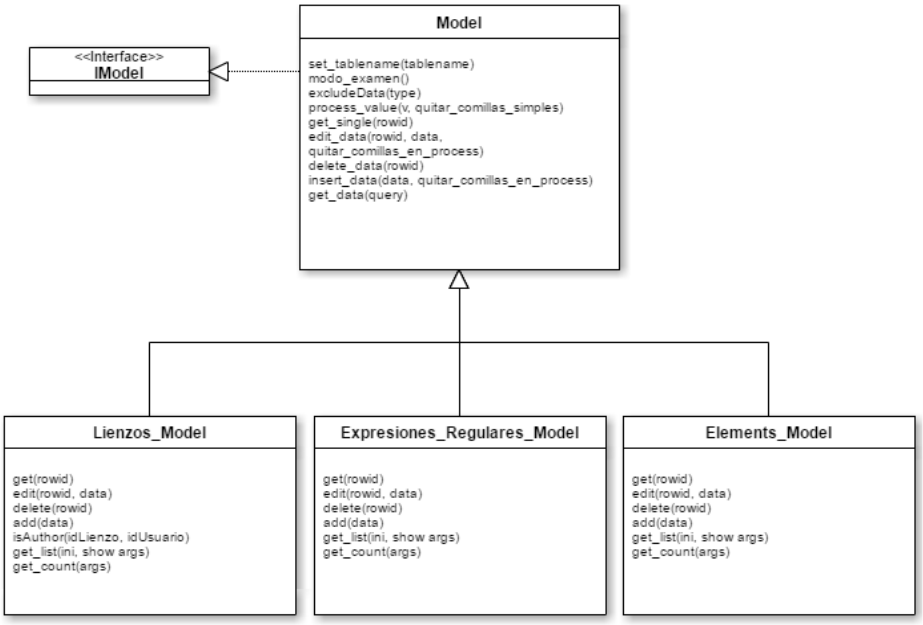

Ilustración 4: Diagrama de clases Modelos

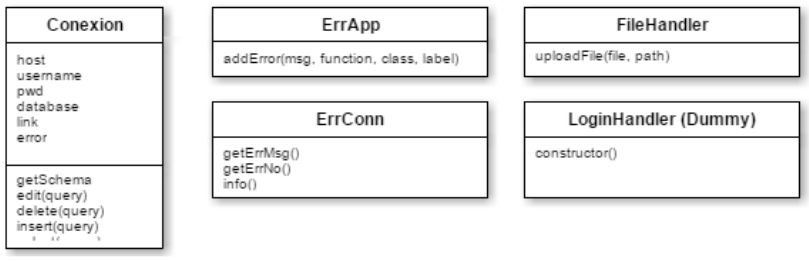

Ilustración 5: Diagrama de clases (Otros)

Tal y como puede verse en las ilustraciones anteriores, para la realización de la aplicación se ha utilizado un patrón MVC. Gracias a la utilización de este patrón, el código ha sido separado en tres capas diferentes acotadas por su responsabilidad:

- **Modelos**: Los modelos se encargarán de la lógica de la base de datos (insert, delete, updates, etc). En este caso nuestros modelos son **(Ilustración 4):**
    - Lienzos Model
    - Expresiones Regulares Model
    - Elements Model
- **Vistas**: Las vistas son todos los HTML/PHP que contiene nuestra aplicación y que mediante la utilización de AJAX realiza peticiones a nuestro servidor. En este caso las peticiones las recibe una clase llamada admin-ajax que se encarga de recoger las peticiones y enviarlas al controlador correspondiente.
- **Controladores**: Los controladores se encargarán de desarrollar la lógica de negocio. Para la aplicación se han desarrollado los siguientes controladores **(Ilustración 3)**:
    - Lienzos Controller
    - Elementos Controller
    - Expresiones Regulares Controller

Por otro lado, tenemos otras clases auxiliares que han sido creadas durante el desarrollo de la aplicación. Estas clases son **(Ilustración 5):**

- Conexión: Se encarga de la conexión con la base de datos.
- ErrApp: Se encarga del manejo de los errores de la aplicación.
- ErrConn: Se encarga del manejo de los errores de la base de datos.
- FileHandler: Se encarga de la subida de ficheros.
- LoginHandler: Se ha preparado un Dummy que permite cambiar el rol de Profesor a Alumno. Esta lógica será implementada por otra persona.

En el siguiente capítulo entraremos más en detalle sobre la implementación de todas estas clases y cómo interactúan entre ellas.

5. Implementación

La implementación se ha llevado a cabo en seis iteraciones, dando como resultado un entregable por cada iteración realizada.

En esta sección detallaremos el desarrollo realizado en cada iteración, así como los problemas surgidos en ellas. Además, incluiremos diagramas de secuencia para explicar con mayor claridad lo que hace la aplicación, lo que facilitará la comprensión de las soluciones planteadas.

Iteración	Tareas
1ª Iteración	• Diseño del esquema de la BBDD. • Diseño y creación de las clases PHP necesarias para gestionar la aplicación.
2ª Iteración	• Diseño de la UI general. Menú y framework AdminLTE.
3ª Iteración	• Diseño de la UI para la creación y gestión de expresiones regulares • Implementación del controlador PHP que gestiona las expresiones regulares.
4ª Iteración	• Diseño de la UI para la creación y gestión de componentes • Implementación del controlador PHP que gestiona los componentes.
5ª Iteración	• Diseño de la UI para el panel de ejercicios y creación del objeto Core en JavaScript. • Implementación del controlador PHP que gestiona los lienzos creados por los usuarios.
6ª Iteración	• Diseño de la UI para la corrección y gestión de los lienzos de los alumnos

5.1 Primera iteración: BBDD y Clases PHP

5.1.1 Diseño del esquema de la BBDD

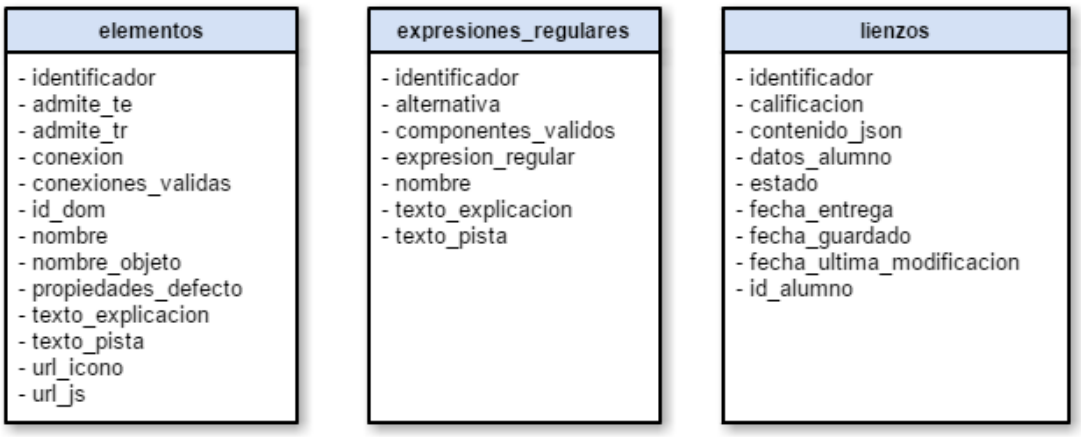

Ilustración 6 - Tablas de iRedes

Para la aplicación hemos creado tres tablas que contienen la información sobre los elementos, las expresiones regulares y los lienzos realizados por los usuarios.

5.1.1.1 Tabla Elementos

Contiene la información sobre los componentes o elementos que conformarán nuestros diagramas. Para cada elemento hemos guardado la siguiente información:

- **identificador**: Número entero auto-incremental que identifica al elemento.
- **admite_te**: Indica si el elemento admite o no tabla de encaminamiento. Esta propiedad será usada a posteriori para realizar las pruebas de validación del lienzo creado por un usuario.
- **admite_tr**: Indica si el elemento admite o no que se le acoplen tarjetas de red.
- **conexiones_validas**: Indica los elementos con los que puede establecer una conexión.
- **id_dom**: Será el prefijo que utilice el Core para ponerle como nombre al objeto en el DOM.
- **nombre**: Nombre que deseamos ponerle al componente.
- **nombre_objeto**: Esta propiedad debe contener exactamente el mismo nombre que el que tiene la variable del objeto contenido en el código JavaScript (js). Por ejemplo, si llamamos al componente Concentrador el fichero js asociado debe contener una variable llamada “Concentrador”. En caso contrario, el Core no sabrá vincular el objeto que deseamos crear.
- **propiedades_defecto**: Son las propiedades que un componente tendrá por defecto al ser añadidas al DOM. Estas propiedades no contendrán valor pero ayudarán al alumno a saber cuáles son las propiedades correctas. Durante el modo de examen no se aplicará.
- **texto_explicacion**: Texto que pretende explicar al alumno en qué consiste el componente añadido. Esta información no aparece durante el modo examen.

- **texto_pista**: Es un texto de pista que le aparecerá al usuario en caso de que no pase la validación y que pretende ayudarle a corregir los errores. Esta información no aparece durante el modo examen.
- **url_icono**: Contiene la URL del icono que representa al componente.
- **url_js**: Contiene el JavaScript que describe el funcionamiento del componente.

5.1.1.2 Tabla expresiones_regulares

Contiene las expresiones regulares que definen las propiedades que pueden ser añadidas a cada uno de los elementos. Cada expresión regular debe contener la siguiente información:

- **identificador**: Número entero auto-incremental que identifica a la expresión regular.
- **nombre**: Es el nombre de la propiedad que luego podrá ser enlazada al componente. El usuario deberá poner este nombre para pasar la validación de su lienzo de manera correcta.
- **alternativa**: Son nombres alternativos para dar mayor flexibilidad a la hora de definir las propiedades en los componentes que conforman el lienzo. De esta manera, un usuario podría pasar la validación si escribe en el nombre de la propiedad un texto existente en estas alternativas. Por ejemplo, si se ha declarado el nombre TCP IP podría declararse como alternativa TCP/IP. Si un usuario escribiese como nombre de la propiedad cualquiera de los dos textos, pasaría la validación si el valor es correcto para la expresión regular.
- **componentes_validos**: Define para cuáles de los componentes dados de alta en el sistema es válida esta expresión regular. Por defecto en el listado aparecerá la opción “cualquiera” que guardará en BBDD un -1 e indicará que la propiedad puede ser usada en cualquier componente.
- **expresión_regular**: Es la expresión regular utilizada para validarla.
- **texto_explicacion**: Es un texto que se ofrece al usuario cuando no se pasa el test de validación, pero sí ha escrito correctamente su nombre.
- **texto_pista**: Es la pista que se ofrece al usuario cuando no pasa el test de validación, pero sí ha escrito correctamente su nombre.

5.1.1.3 Tabla Lienzos

Contiene los lienzos creados por los usuarios, tanto profesores como alumnos. Cada lienzo contiene la siguiente información:

- **identificador**: Número entero auto-incremental que identifica el lienzo.
- **calificacion**: Nota que se le ha dado al lienzo. Solo un profesor puede asignar una calificación a un lienzo.
- **contenido_json**: Contiene el JSON que describe el lienzo con sus componentes, propiedades, conexiones, etc.
- **datos_alumno**: Datos del usuario que ha creado el lienzo.

- **estado**: Indica si el lienzo está entregado o es privado. Hasta que no está entregado el profesor no puede ver el lienzo.
- **fecha_entrega**: Fecha en la que fue entregado el lienzo.
- **fecha_guardado**: Fecha en la que se realizó el primer guardado del lienzo
- **fecha_ultima_modificacion**: Fecha de la última modificación del lienzo.
- **id_alumno**: Identificador del usuario propietario del lienzo.
- **comentario_profesor**: Es el comentario que ha realizado un profesor sobre el lienzo.

5.1.2 Diseño y creación de las clases PHP

Para la realización del proyecto hemos realizado una implementación en la que hemos usado el patrón de arquitectura MVC separando así los datos y la lógica de negocio de la aplicación de la interfaz de usuario. De esta forma, tenemos:

- Dos Interfaces:
  - **IController**: Que contiene la especificación de la interface que implementará nuestra clase Controller de la que extenderán todos nuestros controladores.
  - **IModel**: Que contiene la especificación de la interfaz que implementará nuestra clase Model de la que extenderán todos nuestros modelos.
- Una clase **Controller**: Que contiene la implementación de IController
- Cuatro clases que extienden de Controller:
  - **Elementos_Controller**: Controlador para los elementos de la aplicación.
  - **Expresiones_Regulares_Controller**: Controlador de las expresiones regulares que usan los elementos de la aplicación.
  - **Lienzos_Controller**: Controlador de los lienzos realizados por los usuarios.
  - **Usuarios_Controller**: Controlador de los usuarios con acceso al sistema.
- Una clase **Model**: Que contiene la implementación de IModel
- Cuatro clases que extienden de Model:
  - **Elementos_Model**: Modelo de los elementos de la aplicación
  - **Expresiones_Regulares_Model**: Modelo de las expresiones regulares que usan los elementos de la aplicación.
  - **Lienzos_Model**: Modelo de los lienzos realizados por los usuarios
  - **Usuarios_Model**: Modelo de los usuarios con acceso al sistema.

Además, nuestra aplicación, cuenta con los siguientes ficheros:

- **admin-ajax**: Fichero que se encarga de procesar todas aquellas peticiones Ajax realizadas por la aplicación al servidor.
- **defines**: Incluye todas las definiciones php utilizadas en la aplicación.
- **globals**: Incluye todas las variables globales utilizadas por nuestra aplicación.

- **init_clases**: Contiene diversas funciones utilizadas por nuestra aplicación tales como **_t(), add_action() o do_action()** de las que hablaremos más adelante. Además, contiene todas las instancias de las clases que necesita nuestra aplicación para funcionar.
- **Conexion**: Esta clase se encarga de la configuración de la BBDD y contiene las operaciones elementales realizadas sobre la BBDD (Insert, delete, update, select). La clase Model la utiliza cuando crea un objeto de la clase Conexion().
- **ErrApp**: Clase que se encarga de manejar todos los errores de la aplicación para poder informar correctamente a la vista.
- **ErrConn**: Clase que se encarga de manejar todos los errores de la BBDD para poder informar correctamente a la vista.
- **FileHandler**: Clase para la subida de ficheros.
- **LoginHandler**: Clase para la realización del login. Esta clase deberá implementarse cuando se vaya a realizar la conexión con Siette.

5.1.2.1: Fichero admin-ajax.php

Para la realización de la aplicación se ha realizado una conexión cliente-servidor utilizando AJAX. Para ello hemos creado el fichero admin-ajax que será la que se encargue de procesar todas las peticiones AJAX que nuestra aplicación realice al servidor.

En primer lugar estudiaremos la estructura básica de cualquier petición Ajax realizada por nuestra aplicación al servidor.

```
$.ajax({
    url: "../request/admin-ajax.php",
    type: "GET",
    data: {
        action: "obtener_lista_lienzos",
        args:{
            page: 0,
            show: 10
        }
    },
    dataType: "json",
    success: function(result){
        if(result.success){
            ...
        }else{
            ...
        }
    }
});
```

Ilustración 7 - Ejemplo de llamada AJAX

En la imagen anterior podemos ver una petición para obtener la lista de lienzos. Lo primero en lo que nos fijamos es que la URL no es el EndPoint sino que siempre se realiza a admin-ajax.php. Esto es porque, como hemos dicho, la clase admin-ajax se

encargará de procesar todas las peticiones realizadas por nuestra aplicación al servidor.

Si nos fijamos en la propiedad data de la petición Ajax, veremos que es ahí donde se indica realmente el EndPoint al que queremos llamar y los parámetros que vamos a pasarle. En este caso sería una petición para obtener los primeros 10 lienzos registrados en la base de datos.

Para que esto funcione correctamente en el fichero init_clases hemos definido las funciones:

- **add_action($action_name, $params)**: Esta función nos ayudará a definir cuáles son los **EndPoints** registrados en nuestra aplicación.
- **do_action($action_name, $params=array())**: Esta función ejecutará la correspondiente al **EndPoint** que haya invocado nuestra aplicación. En el ejemplo anterior la llamada que se haría sería algo como:
  - do_action("obtener_lista_lienzos", {$page =>0, $show=>10});
- **_t($text, $output=true)**: Esta función nos ayudará con la internacionalización.

5.1.2.1.1: Función add_action:

Como se ha explicado anteriormente, esta función se encargará de registrar todas aquellas operaciones que tendremos disponibles a modo de **EndPoint** para llamar desde nuestra aplicación.

Para registrar un **EndPoint**, en el constructor de cualquier clase que extienda de Controller, deberemos poner lo siguiente:

```
add_action("obtener_lista_lienzos", array(__class__, "obtenerListaLienzos"));
```

Ilustración 8 - Ejemplo de invocación a función add_action

De esta manera lo que estamos indicando es que, para el EndPoint obtener_lista_lienzos tendremos que ejecutar la función obtenerListaLienzos de la clase en la que se ha declarado el add_action.

La definición de esta función sería tan sencilla como la siguiente:

```
function add_action($action_name, $params){
    global $actions;
    $actions[$action_name] = $params;
}
```

Ilustración 9 - Implementación función add_action

Con ella lo que estamos haciendo es registrar dentro de la global **$actions** el **EndPoint** obtener_lista_lienzos, indicándole en los parámetros cuál será la clase y

el método dentro de la misma que hay que ejecutar cuando se haga un **do_action** de ese **EndPoint**.

5.1.2.1.2: Función do_action

Así como la función add_action registraba los EndPoints, do_action los ejecutará.

Para ello buscará si existe dentro de la variable global $actions una clave con el EndPoint y de ser así tan solo tendrá que instanciar un objeto de esa clase y ejecutar la función indicada con los parámetros deseados.

Esta implementación queda de la siguiente manera:

```
function do_action($action_name, $params = array()){

        global $actions;
        if (is_array($actions) && array_key_exists($action_name, $actions)){
            $class = null;
            $function = null;
            foreach($actions as $key=>$action){
                if ($key===$action_name){
                    //Hemos encontrado los datos
                    $class = $action[0];
                    $function = $action[1];
                    break;
                }
            }

            if (isset($class, $function)){
                $object = new $class;
                if (isset($params["args"]) && !is_null($params["args"])){
                    return $object->$function($params["args"]);
                }else{
                    return $object->$function(array());
                }

            }
        }
}
```

Ilustración 10 - Implentación función do_action

En este método podemos ver cómo, si se ha encontrado el EndPoint, las variables $class y $function, contendrán una instancia de la clase que necesitamos para poder ejecutar la función ($function). Además, si así lo deseamos, podremos pasarle los parámetros indicado en $params de la llamada do_action.

La función do_action devuelve tal cual el resultado de haber ejecutado la función deseada. En nuestra aplicación, esta será siempre un JSON devuelto por la función json_response_print que implementa la clase Controller.

En caso de que no exista un EndPoint con esa clave se producirá un error silencioso.

5.1.2.2: Clases Controller y Model

Una vez tenemos especificada la clase admin_ajax que nos permite realizar las llamadas Ajax desde nuestra aplicación cliente al servidor podremos definir los controladores y modelos que definirán nuestra aplicación.

5.1.2.2.1 Clase Controller

La clase Controller implementa IController y será de la que extiendan todos nuestros controladores.

Controller implementa todos aquellos métodos que nos permitirán retornar el JSON con la información necesaria para informar a nuestra aplicación tras haber realizado la ejecución de un EndPoint.

La clase tiene las siguientes propiedades:

- **$model**: Será el modelo que estemos utilizando en un momento dado desde nuestro EndPoint para trabajar con él (obtener datos de la BBDD, borrar, actualizar, etc.).
- **$errores**: Nos servirá para obtener la información sobre los diferentes errores que pudieran surgir durante la ejecución de un EndPoint.
- **$json_response**: Estructura JSON que devolveremos tras realizar la ejecución de un EndPoint.

Con el fin de mantener una homogeneidad en las respuestas que el servidor da a la aplicación cliente, se han definido una serie de funciones que nos permiten manejar y alterar el objeto json_response. Estas funciones son:

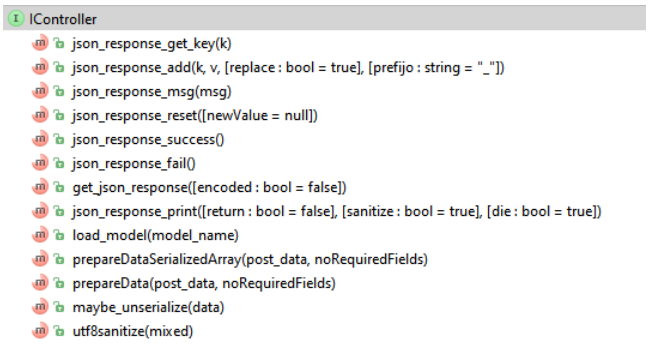

Ilustración 11 - Métodos definidos en la Interfaz IController

Donde:

- **json_response_get_key**: Nos devuelve el valor de una clave dentro del objeto json_response.
- **json_response_add**: Añade una clave al objeto json_response
- **json_response_msg**: Especifica el mensaje que se mostrará al usuario tras ejecutar el EndPoint.
- **json_response_reset**: Volverá a poner en estado inicial el objeto json_response. En caso de pasarle en la variable newValue algo json_response pasaría a valer eso.
- **json_response_success**: Pone success a true. Success será el que en la vista nos dirá si la operación se ha realizado con éxito o no. En caso de error la aplicación cliente sabe que debe mirar los errores para mostrarlos al usuario y actuar en consecuencia.
- **json_response_fail**: Pone success a false.
- **get_json_response**: Nos devuelve el objeto json_response. Se le puede indicar que nos lo devuelva como objeto o que nos lo devuelva ya como un json. Para eso se utiliza el parámetro $encoded.
- **json_response_print**: Será la encargada de finalizar el EndPoint devolviendo la estructura de json_response.
- **load_model**: Cargará el Modelo correspondiente para su uso.
- **maybe_unserialize**: Intenta des-serializar un objeto de base de datos. En caso de no poder realizarse la deserialización devolvería el objeto tal cual. De esta manera, cuando realizamos una consulta a la base de datos, y alguno de los datos está serializado, podemos devolverlo como un objeto.

Otras funciones como **utf8sanitize**, **prepareData** o **prepareDataSerializedArray** nos ayudarán a manejar los datos serializados que se encuentran en la base de datos o que nos envía una petición AJAX mediante GET desde el cliente, pudiendo transformar esa información en un objeto con el que trabajar de manera más sencilla.

Por ejemplo, si se ha realizado una petición AJAX usando el método GET del protocolo HTTP, y cuyos datos se envían mediante la función javascript serialize, al servidor nos llegaría algo parecido a esto:

?variable1=valor1&variable2=valor2&…..&variableN=valorN.

Lo que haría prepareDataSerializedArray es transformar esta información en un objeto como:

```
Array{
        variable1=>valor1,
        variable2=>valor2,
        …
        variableN=>valorN
}
```

Lo que sin duda nos permitiría trabajar con esta información mucho más cómodamente, e incluso guardarlo en BBDD como un objeto que quedaría de la manera:

a:n{ i:0:s:9:'variable1',i:1:s:6:'valor1', i:2:s:9:'variable2',i:3:s:6:'valor2',…}

Lo que haría la función **maybe_unserialize** sería justamente el paso contrario que es, en caso de poder realizarse el unserialize, transformar la cadena del ejemplo anterior al objeto original.

5.1.2.2.2 Clase Model

La clase Model será la que implemente las funciones de la Interfaz IModel y se encargará de realizar las operaciones básicas sobre la base de datos.

Las funciones de la clase Model son:

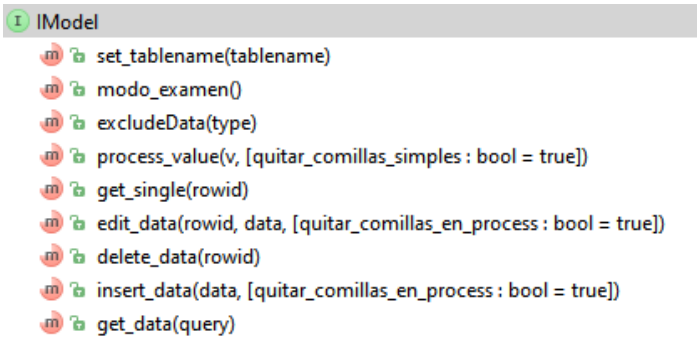

Ilustración 12 - Métodos clase Model

Donde:

- **set_tablename**: Modifica el nombre de la tabla que usará el modelo.
- **modo_examen**: Indicará si el sistema está en modo examen o no.
- **excludeData**: Indica cuáles son aquellos datos que no deben devolverse a la vista. Esto, en conjunción con el modo_examen, servirá para filtrar la información que llega al usuario durante la realización de los lienzos.
- **process_value**: Procesa los datos que van a guardarse en la base de datos para evitar inyección. La clase Model tiene una propiedad llamada allowed_tags que contiene todas aquellas etiquetas html que si se permite guardar en la base de datos.
- **get_single**: Obtiene un registro de la tabla indicada en tablename dado su ID.
- **edit_data**: Edita un registro de la base de datos dado su id y los nuevos datos para ese registro.
- **delete_d**ata: Elimina un registro de la base de datos.
- **insert_data**: Crea un registro en la base de datos. Si $data es un array, podremos crear un insert_bath para añadir múltiples registros en la base de datos.
- **get_data**: Ejecuta una consulta SQL sobre la tabla indicada en tablename.

5.2 Segunda Iteración: UI y Framework AdminLTE.

Durante la segunda iteración se seleccionó un framework de diseño que nos evitase tener que estar creando estilos para la estructura básica de nuestra aplicación como es la cabecera, pie de página, menús, listados, ventanas modales, etc. Para ello hemos seleccionado un framework de diseño llamado AdmiLTE.

AdminLTE es un Framework gratuito y que tan solo necesitaremos descargar y descomprimir en la raíz de nuestro proyecto para tenerlo totalmente funcional. Desde ese momento tendremos disponible todo lo necesario para desarrollar las diferentes partes de la UI.

Básicamente tendremos un menú lateral que contiene las diferentes partes de nuestra aplicación (Expresiones Regulares, Gestor de componentes, Lienzos y Panel de Ejercicios) y una parte central donde se irán cargando las diferentes interfaces desarrolladas para cada uno de las partes del menú.

Además, nos mostrará información sobre el usuario que tiene iniciado la sesión así como la cantidad de expresiones regulares, componentes o lienzos registrados en el sistema siempre en base al usuario autenticado.

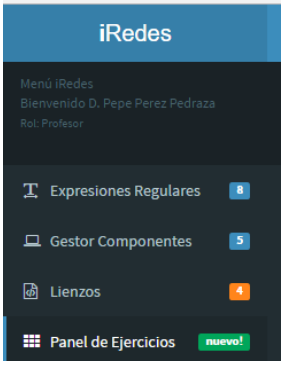

Ilustración 13 - Menú lateral iRedes

5.3 Tercera Iteración: UI Expresiones regulares y su Controlador.

La sección de expresiones regulares consta de cuatro partes:

- Listado de expresiones regulares
- Ventana añadir expresión regular
- Ventana editar expresión regular
- Ventana de validación de expresión regular

Cuyas operaciones las controla la clase Expresiones_Regulares_Controller.php.

5.3.1: Listado de expresiones regulares:

Esta vista consiste en un listado desde la que el usuario podrá realizar las operaciones básicas del CRUD de expresiones regulares: Alta (C), Consulta (R), Actualización (U) y Baja (D), así como la validación de una expresión regular.

Todas las operaciones, se comunican con el servidor a través de una llamada AJAX que procesa admin-ajax y que ejecutará su action correspondiente mediante la función add_action explicada en el capítulo 5.1.2.1.1.

El proceso seguido para la obtención del listado es el siguiente:

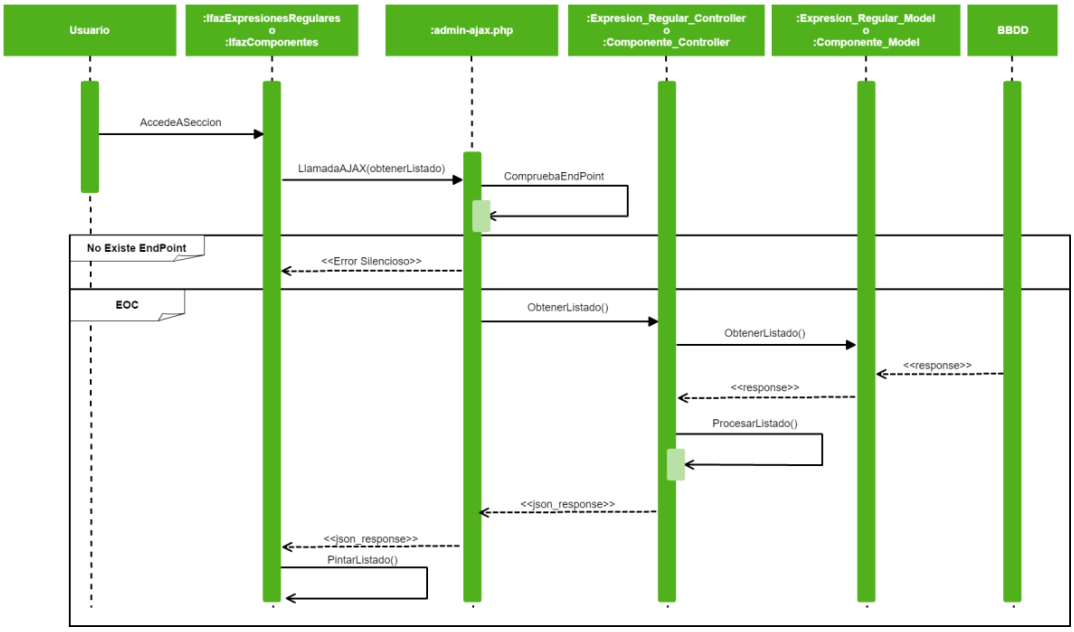

Ilustración 14 - Solicitud de listado (Diagrama de Secuencias)

En el diagrama podemos observar como el usuario actúa de iniciador de la solicitud del listado con tan solo acceder a la sección. Es en ese momento cuando jQuery realizar una llamada AJAX al servidor que procesará admin-ajax con el fin de obtener el listado de expresiones regulares. Aunque lo veremos más adelante, este diagrama es el mismo que se ejecutará cuando se vaya a solicitar el listado de componentes variando tan solo el EndPoint llamado en $.ajax.

5.3.2: Proceso de añadir y Editar expresiones regulares

El proceso interno que se realizan en ambas operaciones está representado mediante los siguientes diagramas de secuencias:

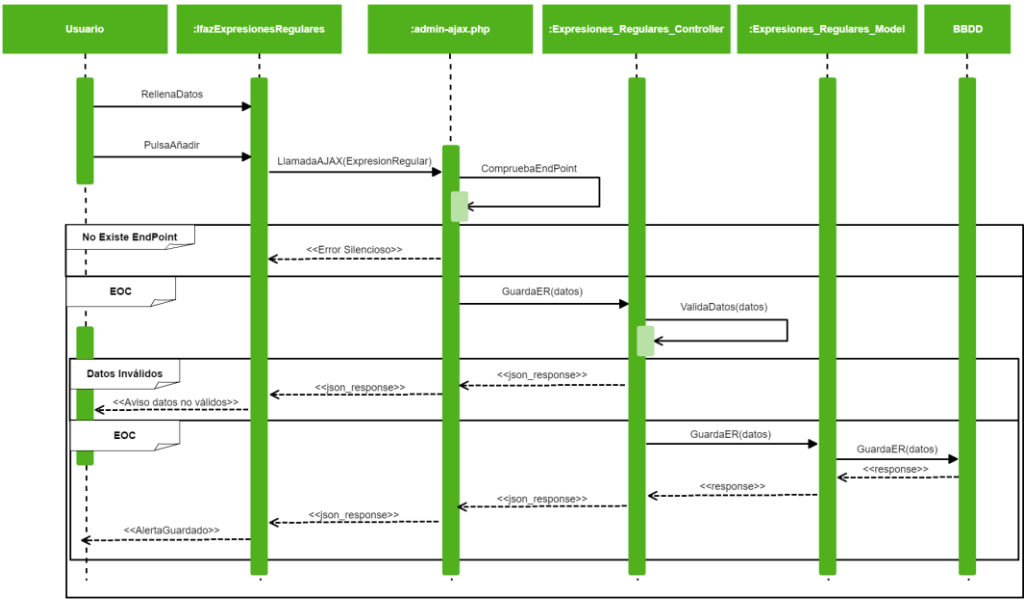

Ilustración 15 - Alta de una Expresión Regular (Diagrama de Secuencias)

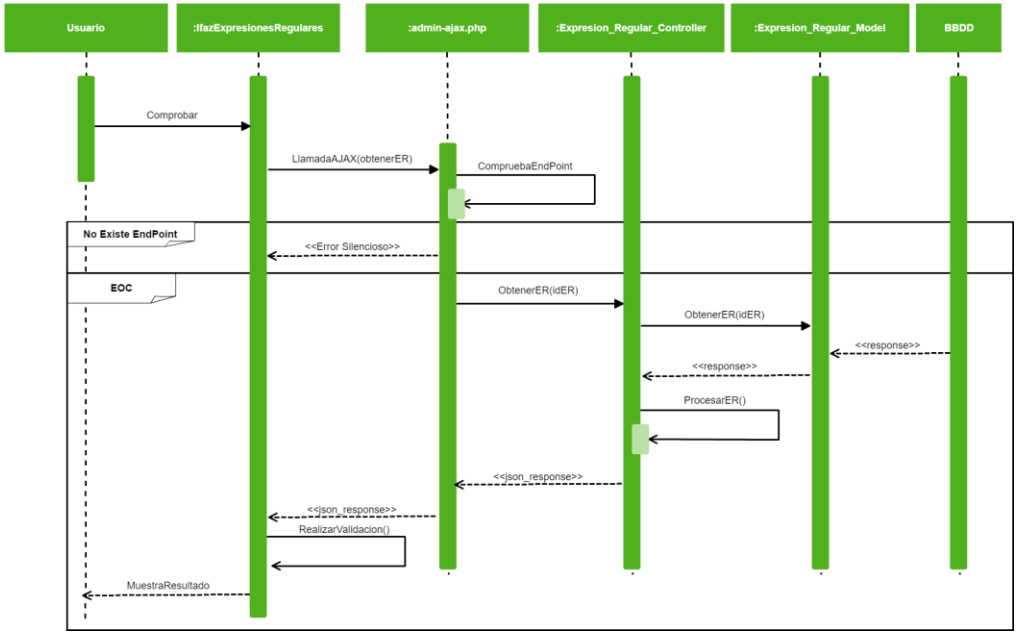

Ilustración 16 - Edición de una Expresión Regular (Diagrama de Secuencias)

En ambos diagramas podemos ver que la secuencia es la misma, ya que todas las llamadas al servidor se realizan de la misma manera y han de pasar por admin-ajax para ser procesadas. Lo único que variará será el EndPoint al que llamaremos (EditaER o GuardaER).

Todas las llamadas AJAX serán devueltas a la vista mediante un JSON para ser procesadas e informar al usuario de lo que ha ocurrido, tanto si se ha producido un error, por ejemplo, en la validación, como si la operación se ha realizado de manera satisfactoria.

Así pues, en los diagramas de secuencias podemos observar los siguientes pasos:

1. El usuario inicia el proceso rellenando los datos y tras eso pulsa el botón Añadir/Editar que inicia la llamada AJAX mediante jQuery enviando los parámetros del EndPoint y los datos a Añadir/Actualizar. Un ejemplo de la llamada lo podemos ver en el punto 5.1.2.1 del presente documento
2. El fichero admin-ajax comprueba la existencia del EndPoint que deber estar registrada mediante add_action previamente. En caso de no existir el EndPoint devolvería un error silencioso y, en caso contrario, ejecutaría la llamada a la función indicada en el action de la llamada AJAX.
3. El controlador recibiría la llamada a la función y validaría los datos antes de añadir/editar el registro. En caso de que no se pase la validación (Falten datos, datos incorrectos, etc) se devolvería ese error a admin-ajax que se lo comunicaría a la vista a través de la promesa de AJAX. En caso de que pase la validación realizará la inserción o actualización del registro en la BBDD y se informaría al usuario del éxito/fracaso de la acción de la misma manera que si se hubiese producido un error, a través de la promesa AJAX.

5.3.3: Validación de expresión regular

Como hemos indicado en el punto anterior el usuario dispondrá de una funcionalidad para comprobar que las expresiones regulares, introducidas en nuestra aplicación, funcionan correctamente.

El diagrama de secuencias para la validación de una ER es el siguiente:

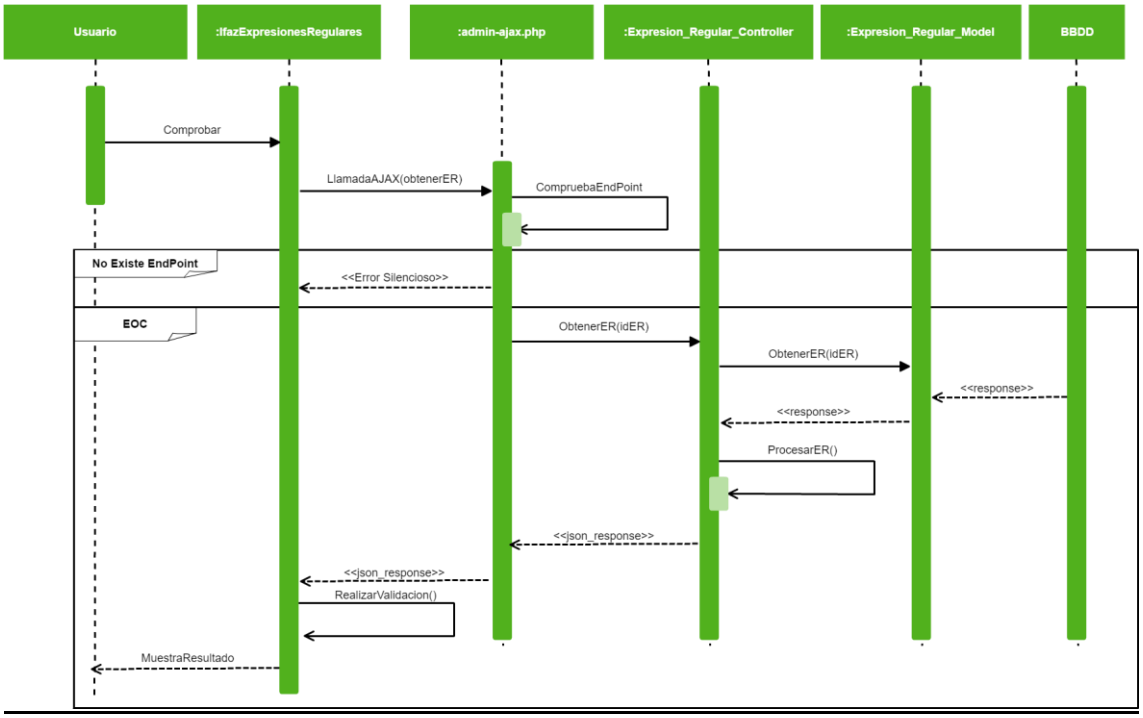

Ilustración 17 - Validación de una Expresión Regular (Diagrama de Secuencia)

En este caso el iniciador es el usuario pulsando el botón Comprobar de la ventana desde la que se validan las expresiones regulares.

De la misma manera que en los diagramas anteriores podemos observar que lo primero que se hace es la comprobar que existe un EndPoint para lo que queremos hacer, que en este caso es obtener los datos de la expresión regular.

Una vez obtenido los datos, se realiza la validación de la misma mediante la función match de jQuery y se muestra al usuario, mediante un cambio de color en la ventana, el éxito o fracaso de la validación.

5.3.4: Controlador Expresiones_Regulares_Controller.php

Expresiones_Regulares_Controller extiende de Controller que implementa la interfaz IController y su estructura es la siguiente:

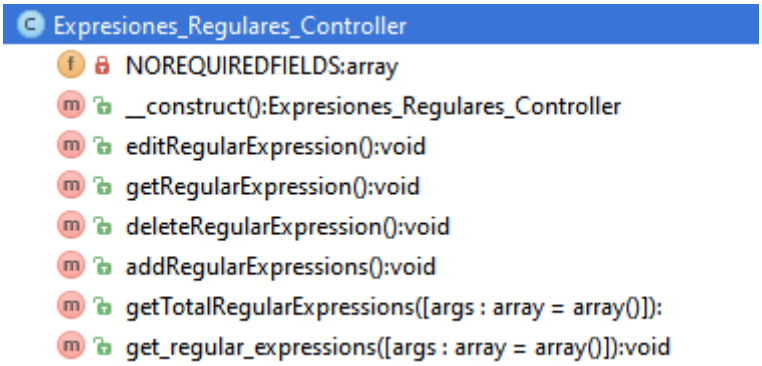

Ilustración 18 - Estructura de la clase Expresiones_Regulares_Controller

Como podemos observar, las operaciones propias que se realizan en esta clase son las necesarias para poder realizar las operaciones del CRUD.

1. editRegularExpresion: Edita una expresión regular. Los nuevos datos vienen a través de AJAX.
2. getRegularExpresion: Se utiliza para obtener una expresión regular en particular. Por ejemplo, se usa para obtener una expresión regular cuando intentamos validarla.
3. deleteRegularExpresion: Elimina una expresión regular. El dato del ID viene a través de AJAX.
4. addRegularExpresions: Permite añadir una expresión regular. Los datos vienen a través de AJAX.
5. getTotalRegularExpresions: Obtiene el número total de expresiones regulares registradas en la base de datos.
6. get_regular_expressions: Obtiene el listado de expresiones regulares registradas en la base de datos.

5.4 Cuarta Iteración: UI Componentes y su Controlador

La sección de componentes consta de tres partes:

- Listado de componentes
- Ventana añadir componentes
- Ventana editar componentes

Cuyas operaciones las controla la clase Expresiones_Regulares_Controller.php.

5.4.1: Listado de componentes

Esta vista consiste en un listado desde el que el usuario podrá realizar las operaciones básicas del CRUD de los componentes: Alta (C), Consulta (R), Actualización (U) y Baja (D) así como consultar el JavaScript que construye el componente.

Como hemos mencionado anteriormente, todas las operaciones, se comunican con el servidor a través de una llamada AJAX que procesa admin-ajax y que ejecutará su action correspondiente mediante la función add_action explicada en el capítulo 5.1.2.1.1.

El proceso seguido para la obtención del listado es el siguiente:

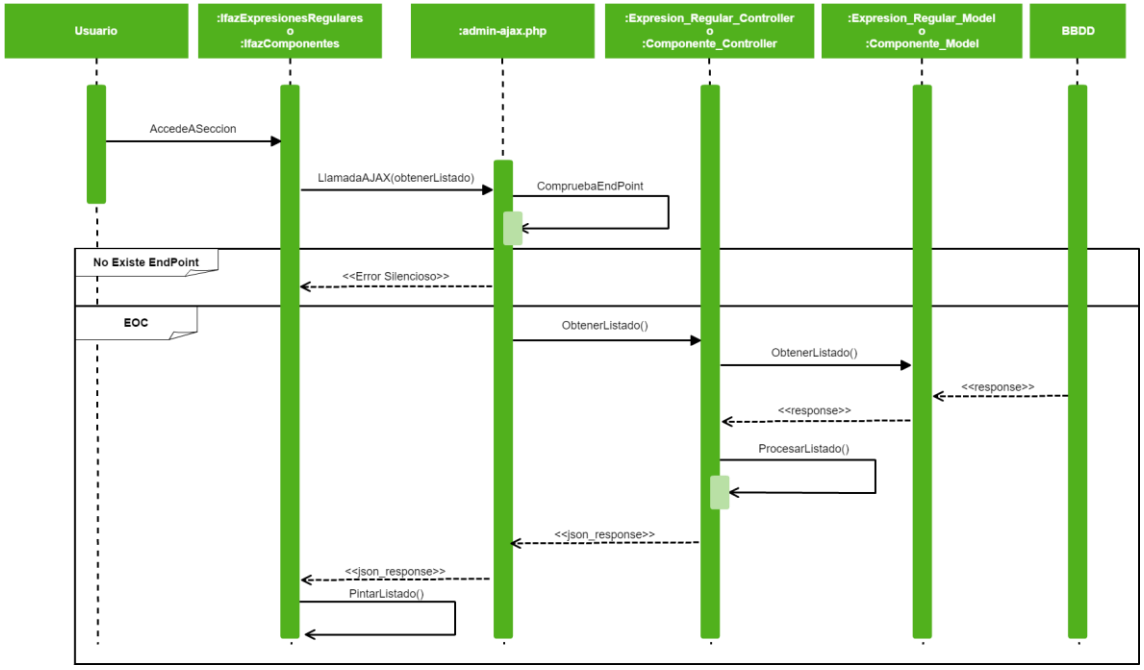

Ilustración 19 - Solicitud de listado (Diagrama de Secuencias)

De la misma manera que hemos visto en el listado de Expresiones regulares, en el diagrama podemos observar como el usuario actúa de iniciador de la solicitud del listado con tan solo acceder a la sección. Es en ese momento cuando jQuery realizar una llamada AJAX al servidor que procesará admin-ajax con el fin de obtener el listado de componentes utilizando para ello la funcion $.ajax.

5.4.2: Proceso de añadir/Editar componentes

El proceso interno que se realizan en ambas operaciones está representado mediante los siguientes diagramas de secuencias:

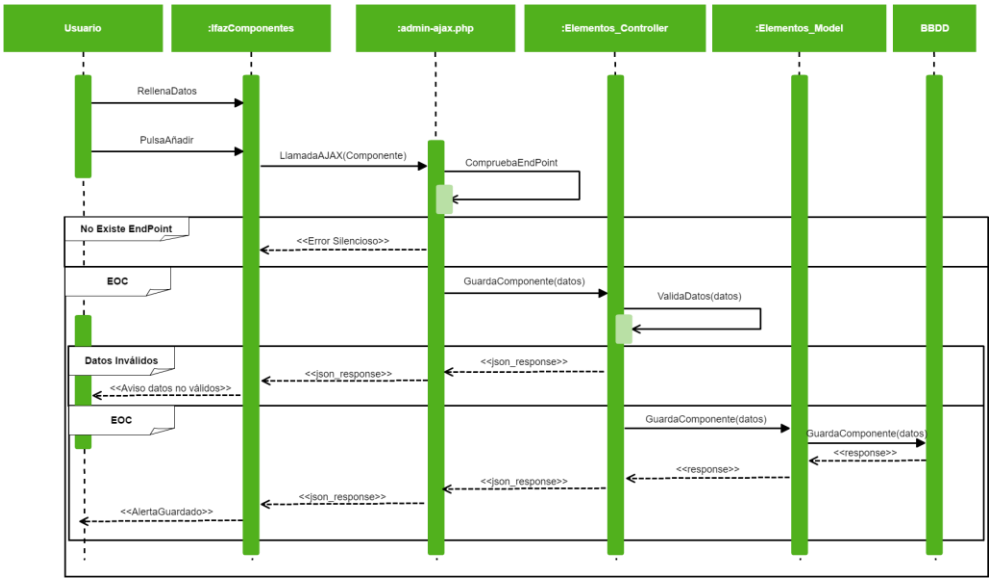

Ilustración 20 - Alta de un componente (Diagrama de Secuencia)

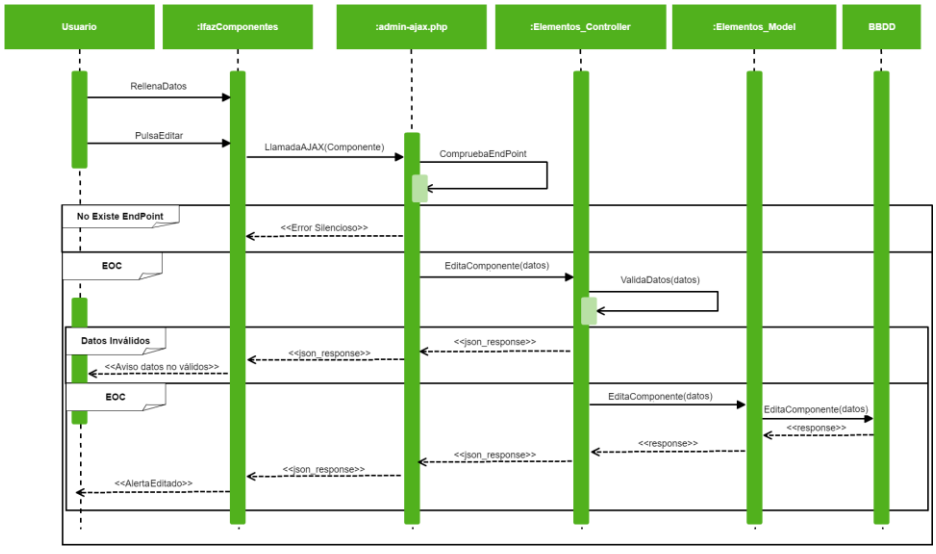

Ilustración 21 - Editar un componente (Diagrama de Secuencia)

Los diagramas son exactamente igual que los vistos en la sección de “Proceso de añadir y editar expresiones regulares” con lo que la explicación de lo que estamos viendo es la misma.

1. El usuario inicia el proceso rellenando los datos y tras eso pulsa el botón Añadir/Editar que inicia la llamada AJAX mediante jQuery enviando los parámetros del EndPoint y los datos a Añadir/Actualizar. Un ejemplo de la llamada lo podemos ver en el punto 5.1.2.1 del presente documento
2. El fichero admin-ajax comprueba la existencia del EndPoint que deber estar registrada mediante add_action previamente. En caso de no existir el EndPoint devolvería un error silencioso y, en caso contrario, ejecutaría la llamada a la función indicada en el action de la llamada AJAX.
3. El controlador recibiría la llamada a la función y validaría los datos antes de añadir/editar el registro. En caso de que no se pase la validación (faltan datos, datos incorrectos, etc.) se devolvería ese error a admin-ajax, que se lo comunicaría a la vista a través de la promesa de AJAX. En caso de que pase la validación realizará la inserción o actualización del registro en la BBDD y se informaría al usuario del éxito/fracaso de la acción, de la misma manera que si se hubiese producido un error, a través de la promesa AJAX.

5.4.3: Controlador Elementos_Controller.php

Elementos_Controller es la clase encargada de realizar todas las operaciones que se ejecutarán desde la vista de Gestor de componentes. Esta clase extiende de Controller que implementa IController.

Los métodos propios de esta clase son los siguientes:

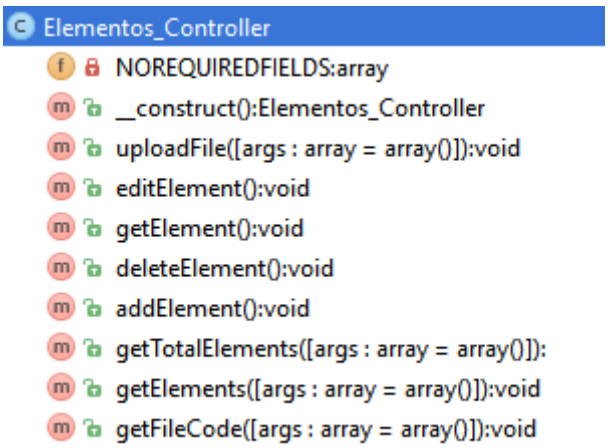

Ilustración 22 - Métodos de Elementos_Controller

1. uploadFile: Se encargará de subir el fichero js y la imagen asociada al componente.
2. editElement: Edita un componente
3. getElement: Obtiene un componente concreto
4. deleteElement: Elimina un componente
5. addElement: Inserta un componente
6. getTotalElements: Obtiene el número total de componentes registrados en la BBDD.

7. getElements: Obtiene el listado de componentes registrados en la BBDD.
8. getFileCode: Obtiene el código js para mostrarlo por pantalla del fichero asociado al componente.

5.5 Quinta Iteración: Panel de Ejercicios y Core.js

Una vez disponemos de los mecanismos para poder registrar los componentes que formarán nuestro lienzo y las expresiones regulares que se añadirán a dichos componentes a modo de propiedades del mismo, podemos crear el core (o núcleo) de nuestra aplicación.

Entre las funciones del core podemos destacar las siguientes:

- Importar de manera dinámica los ficheros javascript de cada componente registrado en la base de datos.
- Clonar los objetos para añadirlos al lienzo añadiendo las propiedades por defecto si es que el componente las tuviera.
- Gestionar el modo examen.
- Realizar las operaciones en base a eventos (onclick, onmove, ondblclick, etc) y realizar los triggers de esos mismos eventos en los componentes (si el componente lo tiene programado en su js)
- Realizar las validaciones de las propiedades de un componente
- Realizar las validaciones generales del lienzo
- Realizar/eliminar conexiones entre los diferentes elementos permitiendo crear diagramas más complejos.
- Permitir la edición de los componentes añadiendo propiedades, tarjetas de red, configurando su tabla de enrutamiento, etc.
- Importar, Exportar, Guardar y Entregar diagramas utilizando JSON.
- Simulación de envío de paquetes

5.5.1: Importación dinámica de ficheros javascript

Lo primero que hace el Core nada más cargar la página es solicitar mediante una llamada AJAX el listado de componentes y expresiones regulares registrados.

Si el sistema se encuentra en modo examen, no solicitará las expresiones regulares y, aunque en los controladores PHP se vacían todos los datos sensibles que aporten información extra al usuario en caso de estar en modo examen, Core, realiza otra comprobación para limpiar las ayudas en caso de que fallase algo en el servidor.

Una vez tenemos disponibles el listado de componentes el Core ejecutará una función llamada cargar_componentes y que recorrerá cada uno de los componentes registrados para ir creando los objetos necesarios para que el usuario pueda incluirlos en el lienzo.

La función cargar_componentes es la siguiente:

```
cargar_componentes: function(){
    /*Cargamos los componentes en el panel lateral y sus respectivos JS*/
    $.each(Core.componentes, function(index){
        var componente = Core.componentes[index];

        /*Añadir aquí los elementos que queramos que tengan por defecto todos los componentes creados*/
        var propiedadesBasicas = {
            config_usuario:{
                propiedades:[],
                conexiones:[],
                tarjetas_red:[],
                tabla_encaminamiento:[],
                comentarios:"",
                posicion:{
                    x:0,
                    y:0
                }
            }
        };

        /*Añadimos las propiedads por defecto a las propiedades básicas*/
        if (jQuery.isArray(componente.propiedades_defecto)){
            $.each(componente.propiedades_defecto, function(index, prop){
                propiedadesBasicas.config_usuario.propiedades.push({
                    descripcion:prop,
                    valor: ""
                });
            });
        }else{
            propiedadesBasicas.config_usuario.propiedades.push({
                descripcion:componente.propiedades_defecto,
                valor: ""
            });
        }

        /*Añadimos las propiedades básicas*/
        $.extend(true, componente, propiedadesBasicas);

        /*Añadimos el icono*/
        $("#componentes-selector").append(
                $("<a/>").attr({
                    "class":"btn btn-app"
                }).append(
                    $("<img/>").attr({
                        "src":componente.url_icono,
                        "class":"img-componente",
                        "data-index":index,
                        "data-type":componente.nombre
                    })
                )
        );

         /*Le pasamos los parámetros
         * Notese aquí que cualquier propiedad que traiga el objeto desde la base de datos será accesible en el componente
         * lo que otorga al sistema de mayor flexibilidad */
         window.common_container[index] = componente;

         //Cargamos su JS
         Core.dynamic_include_js(componente.url_js, "?ver="+Core.version);
    });

    /*Comprobamos si hay un lienzo precargado*/
    if (Core.id_lienzo_precargado>0){
        Core.precargar_lienzo();
    }


},
```

Ilustración 23 - Función cargar componentes

En primer lugar, podemos ver cómo la función crea un objeto llamado componente al que le asignará las propiedades básicas que todo componente debe tener para funcionar. Si ese componente ha sido registrado con unas propiedades por defecto (componente.propiedades_defecto es un array) se las añadiremos al objeto, en caso contrario solo crearemos el objeto con las propiedades básicas.

La función $.extend será la que se encargue de crear un nuevo objeto del tipo componente que posteriormente nos servirá para clonarlo tantas veces como deseemos.

Antes de añadir el fichero js que describe el funcionamiento del componente añadimos una referencia al menú lateral para que el usuario, haciendo clic, pueda añadirlo al lienzo.

Finalmente, mediante la función dynamic_include_js, añadimos el fichero js al DOM para que esté disponible y comprobamos si existe un lienzo para precargar en cuyo caso lo precargaríamos.

5.5.2: Clonación de Objetos al lienzo

Cuando el Core introduce un fichero js de un componente, este fichero js hace una llamada a un método init que debe contener cualquier js.

Veámoslo mejor en un ejemplo:

```
Concentrador = {
    config:{"nombre": ""...},
    init: function () {...},
    set_handlers:function(){
        $("img[data-type='concentrador']").on("click", Concentrador.cloneMe);
    },
    cloneMe:function(){
        var index = $(this).attr("data-index");
        /*Establecemos la configuración recibida para el componente*/
        $.extend(Concentrador.config, window.common_container[index]);

        var clon = $.extend(true, {}, Concentrador);

        /*Hacemos único el identificador del dom con una marca temporal*/
        clon.config.id_dom +=("-"+(new Date().getTime()));

        /*Informamos al núcleo*/
        Core.append_element(clon);

    },
    onMove: function(){...},
    onClick: function(){...},
    onDblClick: function(){...}
}

//Al cargar la página inicializamos el componente
Concentrador.init();
```

Ilustración 24 - Objeto JavaScript (Concentrador)

En el ejemplo estamos viendo la estructura básica que cualquier componente tiene que tener en su javascript.

En ella vemos las funciones:

- Init: Inicializa el componente. Sería como el constructor de una clase.
- Set_handlers: asigna los manejadores necesarios para el comportamiento. Como se puede ver en la imagen lo que estamos haciendo es enlazar el elemento del tipo concentrador añadido por el Core al menú lateral con la función cloneMe del componente.
- cloneMe: Realiza la verdadera clonación del elemento. Mediante la función $.extend creamos un nuevo Concentrador que sería copia exacta del elemento “virgen” tal y como lo creó la función cargar_componentes. Más tarde, la función append_element, se encargará de pintarlo en el lienzo.
- onMove, onClick y onDblClick: Son los manejadores de los eventos asociados al clic, move o onDblClick que se lanzarán desde el Core cuando éste finalice la operación que disparó el evento. No tienen por qué estar definidas si no queremos que el componente se comporte de una manera especial cuando se lancen los eventos.

5.5.3: Resto de funciones destacadas

Como hemos dicho anteriormente existen diversas funciones que se encargan de dar vida a nuestro core pero entrar en detalle sobre todas ellas haría demasiado extenso el documento, por lo que tan solo mencionaremos la función que lo controla y una pequeña descripción de la misma.

Gestión del modo examen:

En las partes críticas de la aplicación comprobaremos mediante la propiedad modo_examen del Core si estamos o no en dicho modo. Esta propiedad se inicializa al obtener el listado de componentes y viene como un atributo del JSON y nos servirá para no dar pistas al usuario durante un ejercicio de examen.

Realizar operaciones en base a eventos click, move, dblClick, etc:

El core dispone de funciones llamadas onClick, onMoveNode, onDblClick, onMouseUp, onMove entre otras. Estas funciones harán lo que estipule el método, como por ejemplo abrir la ventana de propiedades al hacer doble clic sobre un componente o seleccionarlo al hacer clic. Una vez finalice de hacer las operaciones propias del Core comprobará, mediante la función $.isFunction, si en el componente existe ese mismo método y, de ser así, lo invocará.

```
/*Realizamos las acciones individuales del componente al hacer click en él*/
if (jQuery.isFunction(componente.onClick)){
    componente.onClick();
}
```

Realizar las validaciones de las propiedades de un componente (Solo disponible cuando no estamos en modo examen):

La función test_property del Core será la encargada de realizar una validación individual. Esta validación se da cuando el usuario pulsa el icono amarillo que se encuentra al lado de la propiedad introducida en el componente.

Para realizar la validación, tal y como se ha explicado anteriormente, comprobaremos si existe entre las expresiones regulares alguna que tenga como nombre o nombre alternativo la descripción que el usuario ha escrito para la propiedad y, de encontrarse una coincidencia se aplicaría la expresión regular al valor que el usuario ha introducido. Para ello usamos el objeto RegExt de JavaScript y su método test.

```
/*Comprueba expresión regular sobre propiedad*/
check_property: function(property, regular_expression){
    var re = new RegExp(regular_expression);
    return re.test(property);
},
```

Si la propiedad está bien nombrada pero el valor introducido no pasa la expresión regular, se mostrará al usuario un mensaje de error con una pista para esa propiedad. Si ni el nombre ni el valor son válidos, tan solo se dirá al usuario que no ha pasado la validación.

Conectar elementos entre sí:

Lo que nos permitirá realizar un diagrama completo serán las conexiones entre elementos. Además, gracias a esas conexiones, sabremos si el diagrama está bien formado o no (validación de conexiones entre componentes) o realizar envío de paquetes entre elementos.

Para realizar una conexión basta con seleccionar dos o más componentes en el lienzo y pulsar la combinación de teclas SHIFT+Q. Además, para poder realizar la conexión habrá que asegurarse que aquellos componentes que intentan conectarse entre sí disponen de una interfaz física para realizar la conexión, como una tarjeta de red.

Si se seleccionan más de dos elementos la conexión será de todos los elementos con respecto al último que se seleccionó.

La función que se encarga de crear una conexión es Core.crearConexion que se ayudará de un método llamado pintarLineasSVG al que se le pasa el último elemento seleccionado, un listado del resto de los elementos con los que debe conectarse, una distancia con respecto al elemento a conectar y el ángulo de la recta calculados por las funciones calcularDistancia y calcularAngulo respectivamente.

Edición de componentes:

Esta funcionalidad, más que una sola función, es un conjunto de funciones que nos permitirán añadir cada uno de las propiedades al componente, entendiendo por propiedades no solo aquellas que están definidas en las expresiones regulares sino también la lista de conexiones con los diferentes elementos del lienzo, el listado de tarjetas de red, la tabla de encaminamiento, comentarios que un alumno desee realizar sobre el componente que podrá ser visible por el profesor al cargarlo o el mismo nombre del componente dentro del lienzo.

Importar, Exportar, Guardar y entregar diagramas:

Al igual que ocurre con la edición de componentes no se trata de una sola función sino de varias funciones que nos permitirán realizar las operaciones indicadas.

Si bien importar y exportar son tan solo una ventana en la que el usuario podrá copiar/pegar el contenido de un JSON, las operaciones de guardar y entregar diagramas conllevan llamadas AJAX al servidor que guardarán un lienzo o lo marcarán como entregado para su corrección.

Simulación de envío de paquetes:

Para poder realizar la simulación del envío de paquetes, el usuario deberá haber seleccionado dos, y solo dos, elementos del lienzo.

La función que se encargará de realizar esta simulación es Core.mostrarEnviarPaquete que comprobará primero si la configuración de los componentes origen y destino es correcta. Por ejemplo, que tengan dirección IP asignada. Luego intentará realizar el envío del paquete desde el origen al destino pasando por todos los elementos que se encuentren en el camino e indicando el número total de saltos requeridos para el envío.

5.6 Sexta Iteración: UI Corrección de lienzos.

Por último, se ha desarrollado una interfaz que permite al profesor visualizar todos los lienzos, tanto los creados por él mismo como aquellos en los que el alumno haya indicado como entregado el trabajo.

El profesor podrá precargar un lienzo, realizar un comentario sobre el mismo o calificarlo.

Esta interfaz también está disponible para el alumno y podrá visualizar las notas obtenidas, así como los comentarios que un profesor haya realizado sobre el lienzo.

A continuación vemos los diferentes diagramas y sus métodos jQuery:

5.6.1 Precargar lienzo

```
precargar_lienzo: function(){
    $.ajax({
        url: "../request/admin-ajax.php",
        type: "POST",
        crossDomain: true,
        data: {
            action: "precargar_lienzo",
            args: {
                id_lienzo:Core.id_lienzo_precargado
            }
        },
        dataType: "json",
        success:function(result){
            if (result.success){
                if (typeof result.data[0].contenido_json!=="undefined"){
                    var contenido_json = result.data[0].contenido_json.replace(/\[COMILLA_SIMPLE\]/g, "'");
                    $("#json-code-importer").val(contenido_json)
                    Core.importar_json();
                }
            }else{
                Core.show_modal("Error", "No se ha podido precargar el lienzo", "error");
            }
        },
        error:function(xhr,status,error){
            Core.show_modal("Error", error, "error");
        }
    });
},
```

Ilustración 25 - Función precargar_lienzo

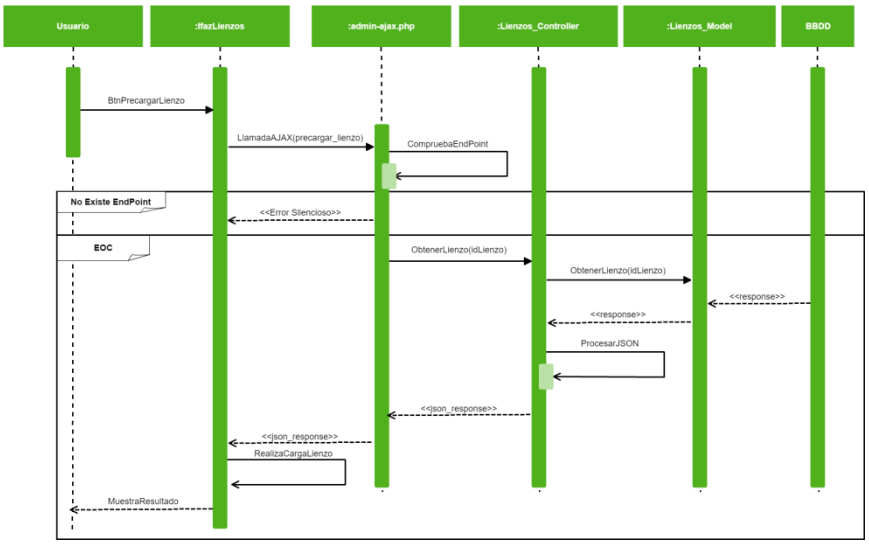

Ilustración 26 - Solicitud de lienzo para precarga (Diagrama de secuencia)

En el diagrama podemos observar, al igual que en cualquier petición AJAX que será admin-ajax la encargada de procesar la petición. Finalmente, lo que se hará será obtener de la base de datos el JSON que conforma el lienzo con el fin de precargarlo en el panel mediante la función Core.importar_json que podemos ver en la ilustración 25.

5.6.2 Comentar y calificar un lienzo

Estas funciones son muy similares, tan solo se diferencian en que en la llamada AJAX en vez del comentario se envía la calificación, así que expondremos tan solo una de las dos a modo de ejemplo.

```
comentar: function(id_lienzo, inputValue){
    $.ajax({
        url: "../request/admin-ajax.php",
        type: "POST",
        crossDomain: true,
        data: {
            action: "comentar_lienzo",
            args: {
                id_lienzo:id_lienzo,
                comentario:inputValue
            }
        },
        dataType: "json",
        success:function(result){
            if (result.success){
                lienzos.show_modal("Comentado", result.message, "success");
                lienzos.getList();
            }else{
                var lineError = $("<tr/>").append(
                    $("<td/>").attr({
                        "colspan":"8",
                        "class"  :"msg-error"
                    }).html("Error: "+result.message)
                );

                $("#listado-lienzos-table").find("tbody").html(lineError);


            }
        },
        error:function(xhr,status,error){
            var lineError = $("<tr/>").append(
                $("<td/>").attr({
                    "colspan":"8",
                    "class"  :"msg-error"
                }).html("Error: "+error)
            );

            $("#listado-lienzos-table").find("tbody").html(lineError);
        }
    });
},
```

Ilustración 27 - Función comentar lienzo

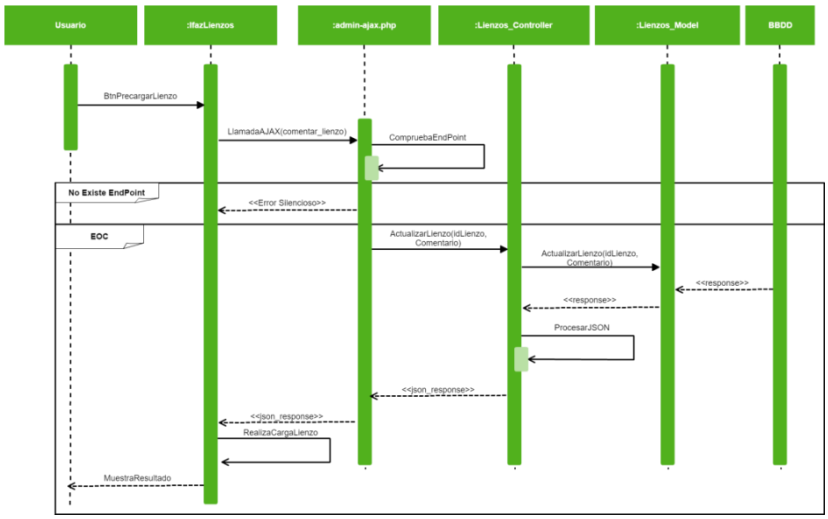

Ilustración 28 - Comentar un lienzo (Diagrama de Secuencia)

En la ilustración 27 podemos observar el código javascript que realiza la llamada AJAX, mientras que en la 28 podemos ver el diagrama de secuencias de esta operación. Como en las demás, el flujo es el mismo devolviendo siempre un JSON con el resultado de la operación realizada.

6: Conclusiones

Uno de los requisitos fundamentales de este TFG era desarrollar una aplicación basada en jQuery que fuese estable, que pudiera ampliarse y mantenerse con facilidad y que permitiese a los alumnos realizar alguno de los ejercicios tipo de la asignatura de redes mediante una plataforma web que fuese fácil de usar.

A título personal creo que se han cubierto completamente estos requisitos por los siguientes motivos:

- El sistema permite la adicción de nuevos componentes permitiendo que el sistema siga creciendo. Además, estos componentes, pueden programarse de manera independiente obteniendo así mayor libertad a la hora de crearlos y dotando al sistema de una mayor flexibilidad.
- El sistema dispone de la posibilidad de realizar validaciones, generar informe sobre las validaciones realizadas, importar/exportar lienzos, guardarlos para continuar con ellos luego, realizar entregas de ejercicios, simular envío de paquetes y corregir lienzos que son las funcionalidades básicas requeridas para este proyecto.
- El sistema dispone de una interfaz clara que permite que el usuario navegue por ella sin perderse y teniendo claro en todo momento en qué sección se encuentra. Dispone de atajos de teclado para que la usabilidad sea mayor y utiliza estilos basados en Bootstrap, lo que de nuevo hace que sea más sencilla de mantener por cualquier persona en un futuro.

A título académico puedo decir que he afianzado mis conocimientos en jQuery, PHP, CSS y aprendido un poco sobre SASS. He visto cómo poder comunicar objetos de jQuery sin tener que utilizar para ello las variables de sessionStorage o localStorage y he podido crear un “miniframework” basado en el patrón MVC para PHP sin necesidad de utilizar alguno de los ya existentes como Codeigniter3, Zend, etc, Este aspecto, por otro lado, ha permitido que la aplicación sea mucho más ligera.

6.1: Posibles mejoras

Como en todo proyecto, la aplicación resultante es mejorable. Durante su desarrollo he notado que, como posibles mejoras, podrían realizarse las siguientes:

- Permitir exportar los diagramas en otros formatos: Como por ejemplo en PDF, JPG, SVG, etc.
- Realización de envíos de correos electrónicos cuando un profesor califique o comente una tarea o cuando un alumno realice una entrega.
- Buscadores en los listados
- Permitir asignar en masa un lienzo realizado por un profesor a los alumnos que vayan a realizar un ejercicio de examen para que no tengan que importarlo a mano.
- Internacionalización de la aplicación. Aunque ya dispone de la función _t() creada para la internacionalización no existe una estructura de ficheros de traducción o la internacionalización de la parte javascript.
- Aumentar el nivel de seguridad en la aplicación mediante la ofuscación de los ficheros javascript para que no puedan ser fácilmente manipulables por los alumnos.

Algunas de estas mejoras que se proponen no se han realizado durante la implementación del proyecto debido a que en sí el proyecto ya era demasiado extenso, pero sería interesante se realizasen.

BIBLIOGRAFIA

PHP: https://secure.php.net/manual/es/
jQuery: http://api.jquery.com/
Sass: http://sass-lang.com/documentation/file.SASS_REFERENCE.html
phpStorm: https://www.jetbrains.com/phpstorm/
MySQL Worbench: https://www.mysql.com/products/workbench/
MySQL: https://dev.mysql.com/doc/
Cacoo Diagramas: https://cacoo.com/
Bootstrap: http://getbootstrap.com/
AdminLTE: https://almsaeedstudio.com/themes/AdminLTE/index2.html

ANEXO I: Manual del usuario

AN1: Navegación.

La navegación de la página se realiza mediante el menú que se encuentra en la parte izquierda.

En dicho menú podemos observar que tenemos las secciones:

- Expresiones regulares
- Gestor de componentes
- Lienzos
- Panel de Ejercicios

La sección de **expresiones regulares**, a partir de ahora ER, será la que se encargue de gestionar todas aquellas ER que deseemos que nuestros usuarios dispongan a modo de propiedades cuando vayan a añadir un componente a su lienzo.

La sección de **Gestor de componentes** será la que se encargue de la gestión de todos aquellos componentes que luego un usuario podrá añadir al lienzo**.**

La sección de **lienzos** será la que se encargue de la gestión de los lienzos que se hayan guardado en la base de datos. Esta sección dependerá del rol del usuario. Mientras que un usuario con rol de profesor podrá, no solo listar y precargar los lienzos guardados propios de los alumnos que hayan entregado una tarea, sino también calificar o comentar dichos lienzos.

Por último la sección de **Panel de ejercicios** será donde los usuarios podrán crear los lienzos usando los componentes y propiedades que se han registrado en la base de datos.

El menú lateral permite colapsarse para poder utilizar más espacio en la pantalla lo que será muy útil durante la creación de lienzos.

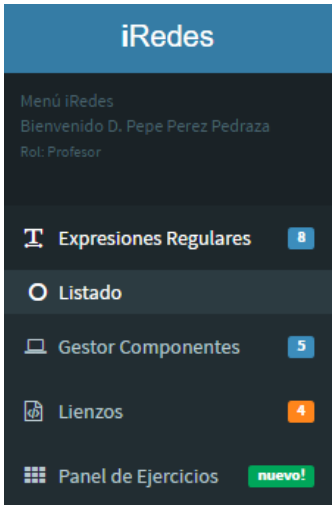

Ilustración 29 - Menú lateral de la aplicación

AN2: Expresiones Regulares

La vista de las expresiones regulares es la siguiente:

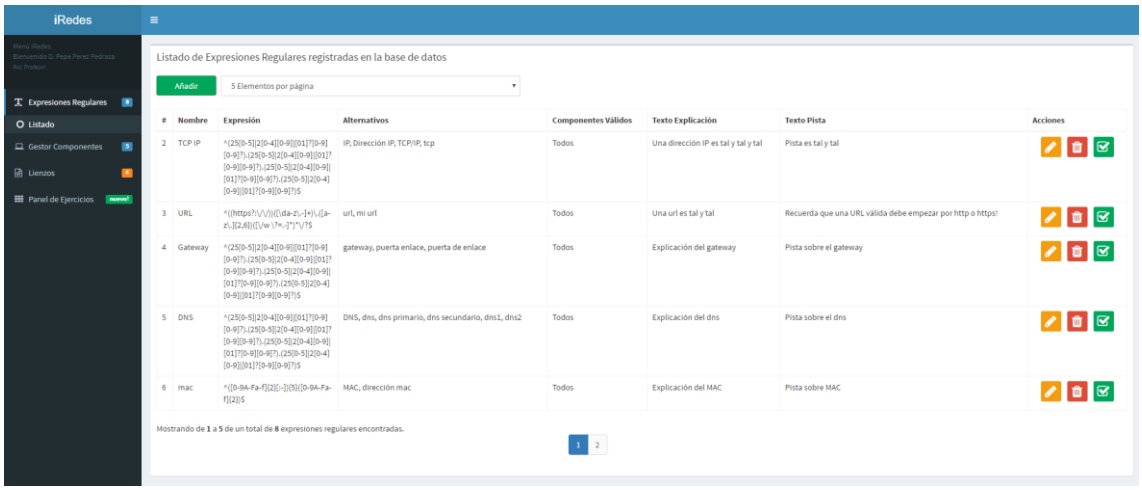

Ilustración 30 - Vista de la sección Expresiones Regulares

En esta vista podemos observar:

- El listado en la parte central
- El filtro de la parte superior nos permite mostrar más o menos registros
- El botón añadir en la parte superior nos permite añadir una nueva expresión regular a nuestra base de datos.
- La paginación en la parte inferior nos permite movernos por las diferentes páginas del listado.
- En la botonera de acciones dispondremos, para cada elemento de la lista:
    - Botón editar: Representado mediante el icono amarillo con el dibujo del lápiz el usuario podrá editar el registro seleccionado.
    - Botón validar: Representado mediante el icono verde con el dibujo del aspa el usuario podrá realizar una validación de la expresión regular.
    - Botón eliminar: Representado mediante el icono rojo con el dibujo de la papelera el usuario podrá eliminar el registro seleccionado. Antes de poder eliminarlo el usuario deberá confirmar la eliminación:

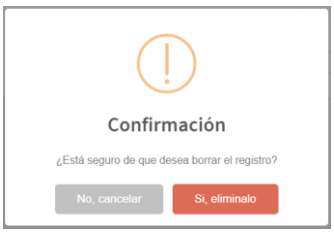

Ilustración 31 - Confirmación de eliminación de un registro.

AN2.1: Añadir y editar una expresión regular

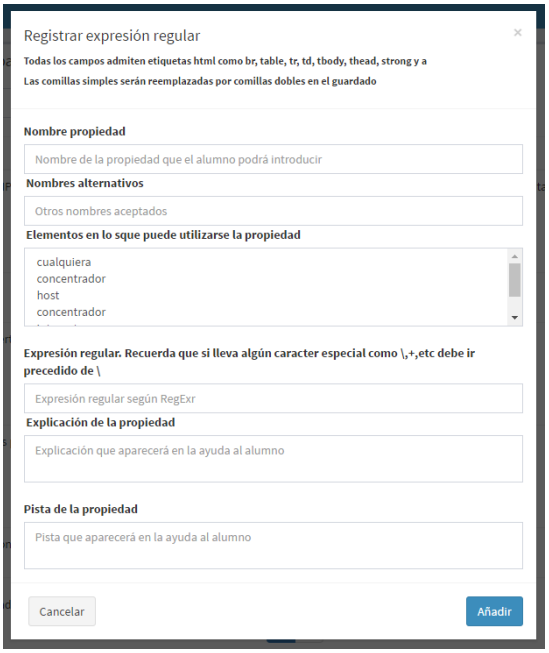

Ilustración 32 - Ventana para Añadir una expresión regular

Los campos que deberá rellenar el usuario son:

- **Nombre propiedad**: Es el nombre principal de la propiedad. Este será el nombre que el sistema cogerá en primer lugar para realizar las validaciones de los lienzos creados por los usuarios.
- **Nombres alternativos**: Estos nombres son otra manera de llamar a la propiedad. Deben definirse escribiendo las diferentes opciones separadas por coma. En caso de que el usuario no escriba tal cuál el nombre principal, el sistema, cogerá de este campo los posibles nombres alternativos para ver si alguno se ajusta a lo que ha escrito el usuario. Esto se explicará en más detalle en la sección en la que explicamos el Panel de Ejercicios.
- **Elementos en los que puede utilizarse la propiedad**: Estos elementos han de ser dados de alta primero y serán en los que esta propiedad serán válidos.
- **Expresión regular**: Es la expresión regular que define la propiedad
- **Explicación de la propiedad**: Es una explicación de la propiedad que aparecerá si el usuario escribe correctamente el nombre pero no pasa la validación. Sólo aparecerá en modo aprendizaje.
- **Pista de la propiedad**: Es una pista que aparece en caso de que el usuario haya asignado un valor erróneo a la propiedad. Sólo aparecerá en modo aprendizaje.
- De la misma manera disponemos de la opción de editar una expresión regular registrada. Para ello el usuario tan solo deberá pulsar el botón de color amarillo con el icono del lápiz. Esto hará que se abra una ventana igual que la de añadir pero con los datos rellenos en los que el usuario podrá modificar aquellos datos que desee. Una vez finalice deberá pulsar el botón salvar.
- Tanto el botón “Añadir” de la ventana de nueva expresión regular como el botón “Salvar” de la ventana de edición, realizan una llamada AJAX. Una vez se ha realizado la llamada AJAX será PHP quien compruebe que los datos que nos llegan

son válidos y de serlos se llevará a cabo la operación de “Añadir” o “Salvar” y en caso contrario mostrará al usuario el mensaje de error correspondiente.

- Por ejp, si el usuario dejase en blanco alguno de los campos obligatorios podríamos ver un mensaje como el siguiente:

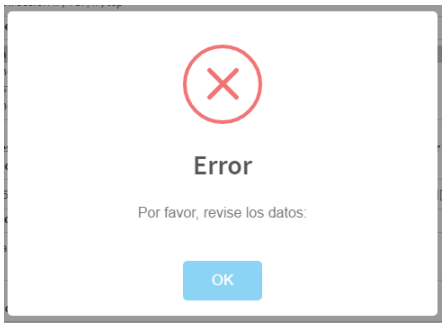

Ilustración 33 - Ventana de error durante la validación de datos.

- Tras lo que el usuario podrá visualizar en la pantalla de “Añadir” o “Editar” cuáles son los datos erróneos:

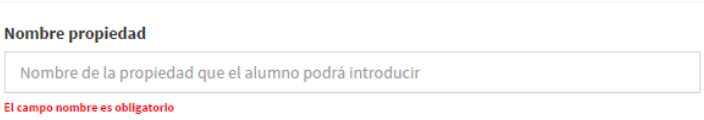

Ilustración 34 - Aviso de campo obligatorio

- En todas las ventanas de alta o edición de nuestra aplicación podremos observar el mismo funcionamiento en cuanto a validación de los campos que se introducen.

AN2.2: Validar una expresión regular (ER)

Para realizar una validación de una ER ya registrada en nuestra aplicación el usuario deberá pulsar el icono verde de la columna acciones en el listado de expresiones regulares.

Una vez pulsado se abrirá una ventana con la expresión regular ya introducida y un cuadro para introducir el valor a validar.

Ilustración 35 - Ventana de validación de ER antes de darle al botón coprobar

En esta ventana el usuario podrá no solo introducir un valor sino realizar modificaciones sobre la ER para depurarla. No obstante estas modificaciones no se hacen permanentes, para ello deberá usar la función editar.

Una vez relleno el usuario deberá pulsar el botón comprobar y el sistema mostrará el éxito o fracaso de la validación mediante el cambio del fondo del cuadro poniéndose verde en caso de que la validación sea exitosa o rojo en caso de fracaso.

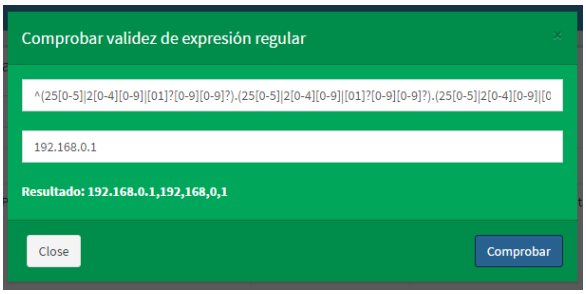

Ilustración 36 - Ejemplo de éxito en validación de una ER

Ilustración 37 - Ejemplo de fracaso en validación de una ER

AN3: Gestor de componentes

La vista de gestor de componentes es la siguiente:

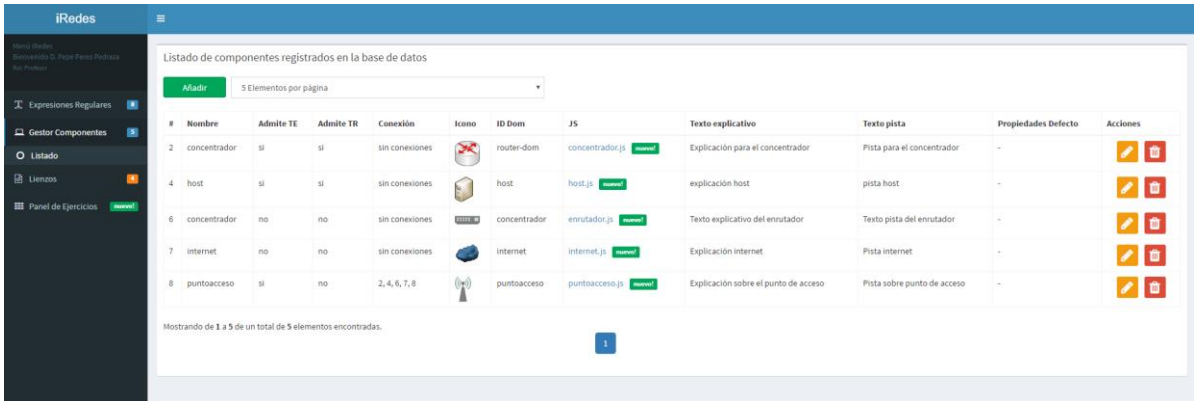

Ilustración 38 - Vista general de la sección Gestor de componentes

En esta sección comprobamos una distribución igual a la de la vista anterior. En la parte superior dispondremos de las acciones de añadir o el filtro para mostrar más o menos registros en la tabla.

En la parte inferior podemos ver la paginación. En el ejemplo solo hay una página ya que solo tenemos 5 componentes metidos.

En la parte central disponemos del listado de componentes que hay registrados hasta el momento. En este caso, además de mostrar simple información, el usuario podrá pulsar sobre el fichero .js para consultar el contenido del dicho fichero (Solo a modo de consulta, no de edición).

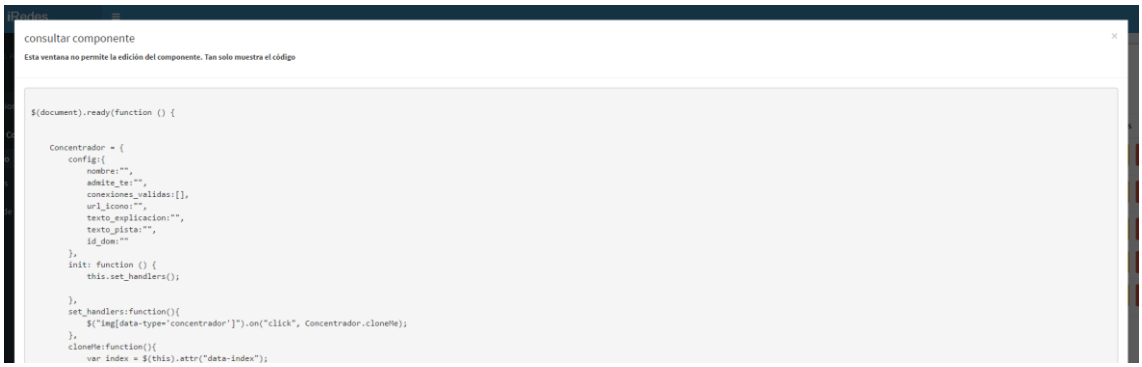

Ilustración 39 - Consultando el fichero js del componente Concentrador

Además, como en todas secciones, dispondremos de las acciones de borrar (Botón rojo) o editar (botón amarillo) en la columna derecha de la tabla.

Si pulsamos sobre el botón borrar nos pedirá una confirmación de borrado. Si el usuario pulsa “Si, elimínalo” el registro se borrará y se avisará al usuario del éxito de la operación. Si se pulsa sobre cancelar igualmente avisará al usuario de que la operación ha sido cancelada.

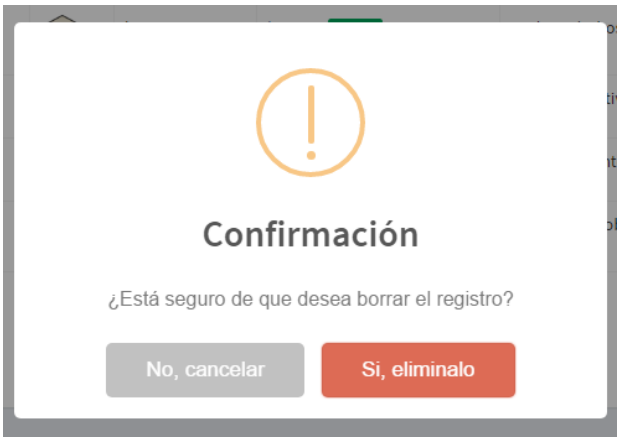

Ilustración 40 - Confirmación de borrado de un componente.

AN3.1 Añadir y editar un componente

Para añadir o editar un componente hay que bien o pulsar el botón añadir que se encuentra en la parte superior del listado o pulsar el botón amarillo (editar) que se encuentra en la columna izquierda de la lista para el componente que deseamos editar.

La ventana que aparecerá en ambos casos es la misma, la diferencia es tan solo la operación AJAX que se realiza y que en la edición la información del componente ya aparece precargado.

En la ventana nos encontraremos primero con la subida la imagen y del fichero js.

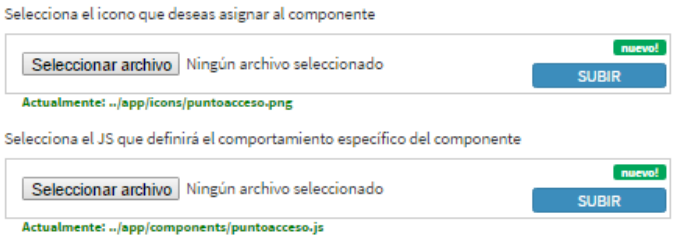

Ilustración 41 - Subir imagen y fichero js asociado al componente.

En el ejemplo de la ilustración 13 podemos ver cómo aparecería durante la edición. La única diferencia con respecto al añadido es que la parte que pone en verde “Actualmente…” no se vería puesto que el componente sería nuevo. En este caso, como decimos, se trata de una edición y esos serían los ficheros asociados al componente, imagen (puntoacceso.png) y javascript (puntoacceso.js).

Un componente tiene que tener a la fuerza asociado una imagen y un js. A diferencia de la edición, el añadido, requerirá que primero se suban estos ficheros. Más tarde se vincularán a la información proporcionada.

El resto de la información, tal y como se puede ver en la Ilustración 14 son las propiedades del componente. Estas propiedades son:

- Nombre: Nombre que usaremos para identificar al componente
- Identificador del DOM: Identificador que tendrá en el HTML. A este ID se le añadirá un timestamp para hacerlo único.
- Indica si admite tabla de encaminamiento: Sirve para que durante la validación se compruebe si el usuario ha introducido una tabla de encaminamiento en un componente que no debe llevarlo.
- Indica si admite tarjetas de red: De la misma manera que la tabla de encaminamiento se validará si el componente debe o no llevar tarjetas de red.
- Selección de los componentes que admiten conexión con este componente. Marca cualquiera para que se pueda conectar con todos.
- Texto explicativo sobre el componente: Este texto aparecerán en las ayudas al alumno durante la creación de lienzos siempre y cuando no esté en modo examen.
- Pista: Texto explicativo a modo de pista que aparecerá al alumno durante la realización de los ejercicios siempre que no esté en modo examen.

- Propiedades: En el último cuadro podrán añadirse aquellas propiedades que se quiera que el componente tenga por defecto cuando el alumno esté realizando un ejercicio que no sea en modo examen.

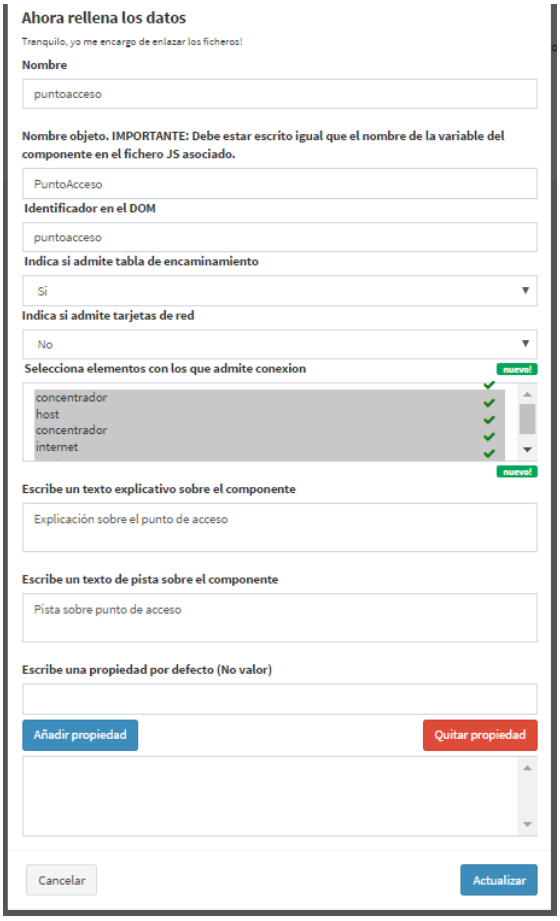

Ilustración 42 - Alta/Edición de un componente. Resto de propiedades

Una vez se haya finalizado de editar o rellenar los campos necesarios el usuario pulsará en el botón azul de la parte inferior derecha (actualizar o añadir en función de la operación que esté realizando).

AN4: Lienzos

Esta sección se distribuye de la misma manera que las anteriores. La única diferencia es que desde aquí no hay un botón añadir y las acciones que pueden llevarse a cabo desde este listado son diferentes.

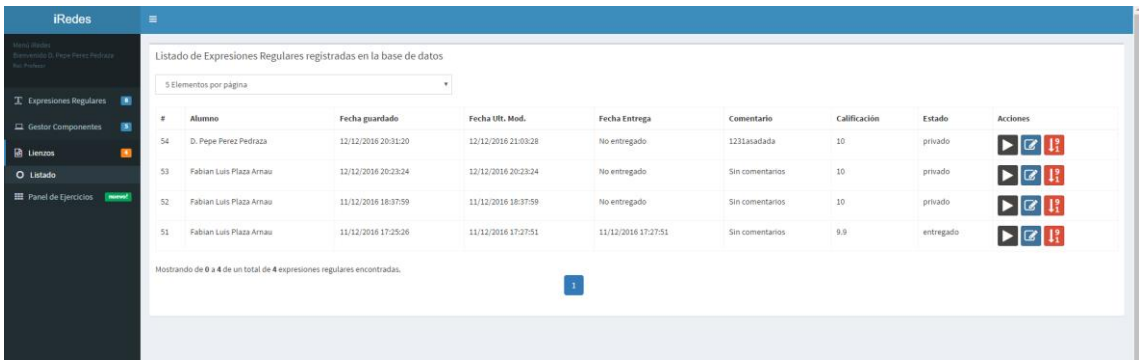

Ilustración 43 - Vista de la sección de lienzo

Las acciones que podemos llevar a cabo desde esta sección son:

- Precargar un lienzo: Cualquier usuario
- Comentar un lienzo: Solo rol profesor.
- Calificar un lienzo: Solo rol profesor.

La operación precargar un lienzo hará que el usuario navegue a la sección de panel de ejercicios donde se cargará el lienzo completo con sus componentes, propiedades y conexiones. Desde ese momento el lienzo está listo para seguir trabajando con él.

El usuario debería ver el siguiente mensaje al realizar la precarga:

Ilustración 44 - Operación precarga exitosa

La operación comentar y calificar un lienzo mostrarán una ventana emergente en la que el profesor podrá calificar o comentar el lienzo.

Ilustración 45 - Ventana comentar lienzo

Tal y como se puede ver en la Ilustración 16, si el usuario intenta guardar un comentario sin haber escrito nada, el sistema le mostrará un aviso indicando que le comentario no puede estar en blanco. De igual manera, si un profesor intenta meter una calificación que no sea una nota numérica entre 0 y 10 (ambos incluidos) el sistema mostraría un mensaje de error para que el profesor rectifique el valor introducido.

Una vez introducido un valor correcto y pulsado el botón OK el profesor recibirá una alerta indicando que la operación se ha realizado con éxito.

Ilustración 46 - Éxito durante la calificación de un lienzo

AN5: Panel de ejercicios

El panel de ejercicios se ve de la siguiente manera:

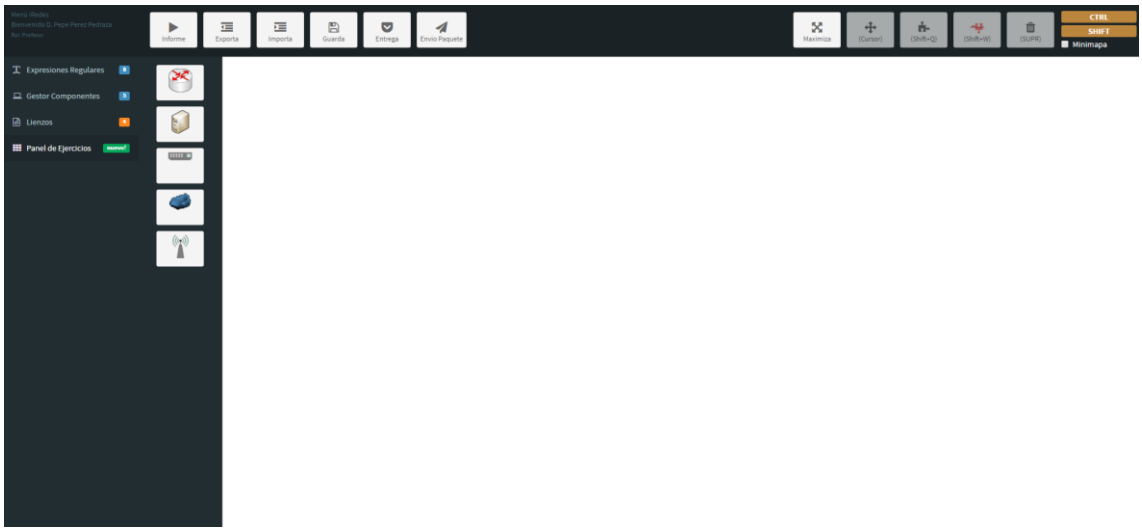

Ilustración 47 - Panel de ejercicios

Este panel contiene 3 partes.

- La parte superior, con la botonera de acciones.
- La parte lateral izquierda, con la botonera de componentes.
- La parte central con el lienzo en blanco donde el usuario podrá ir añadiendo los componentes que desee.

Ilustración 48 - Botonera de acciones

Tal y como puede verse en la Ilustración 20 las acciones que tiene un usuario disponibles durante la creación de un diagrama son:

- Informe: Realiza una validación general del diagrama: Conexiones, Propiedades en los componentes, tarjetas de red, tablas de encaminamiento y muestra un informe sobre los errores producidos en caso de existir.
- Exporta: Realiza la exportación del JSON que conforma el diagrama. Esta exportación aparece en un cuadro de texto con la posibilidad de copiarlo directamente mediante el botón “Copiar al portapapeles”
- Importa: Realiza la importación de un JSON que se pegue en el cuadro de texto dispuesto para ello.
- Guarda: Guarda el Diagrama, no lo entrega. Desde ese momento está disponible en la sección de lienzos y puede ser precargado y modificado cuantas veces se quiera.
- Entrega: Realiza la entrega del diagrama. Desde ese momento aparece en la sección de lienzos y puede ser comentado y evaluado por el profesor.
- Envío de paquetes: Simula el envío de paquetes entre dos nodos del diagrama seleccionados. Debe seleccionarse dos y solo dos elementos y ambos tienen que disponer de tarjeta de red y la propiedad TCP/IP con un valor correcto para poder enviar paquetes entre ellos.

- Maximiza: Permite maximizar el lienzo para que ocupe el 100% de la pantalla y así el usuario disponga de más espacio para poder crear los diagramas.
- Los botones Cursor, Shift+Q, Shift+W o SUPR aparecen deshabilitados porque tan solo informan al usuario del atajo de teclas que permite al usuario mover, unir, desunir o borrar componentes en el diagrama.
- CTRL y SHIFT son tan solo botones informativos que cambian de color cuando el usuario mantiene presionadas esas teclas.
- Minimapa: Muestra un minimapa con la posición de todos los componentes del lienzo así como aquellos que están seleccionados.

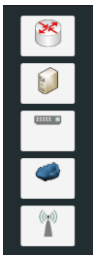

Ilustración 49 - Botonera de componentes

En la Ilustración 21 podemos ver los diferentes componentes que se han registrado en la base de datos y que el Core ha importado. Para añadirlos al lienzo bastará con hacer click sobre uno de ellos y aparecerá en el lienzo disponible para moverlo, conectarlo con otros componentes o editar sus propiedades.

Para arrastrar un componente hay que hacer click sobre él y arrastrarlo hasta donde deseemos lcolocarlo. La posición de este componente dentro del lienzo se actualizará no solo en el minimapa sino también dentro de las propiedades de dicho componente.

Para poder acceder a las propiedades del componente deberemos hacer doble click sobre el componente. Es en este momento en el que podremos editar o añadir propiedades sobre él.

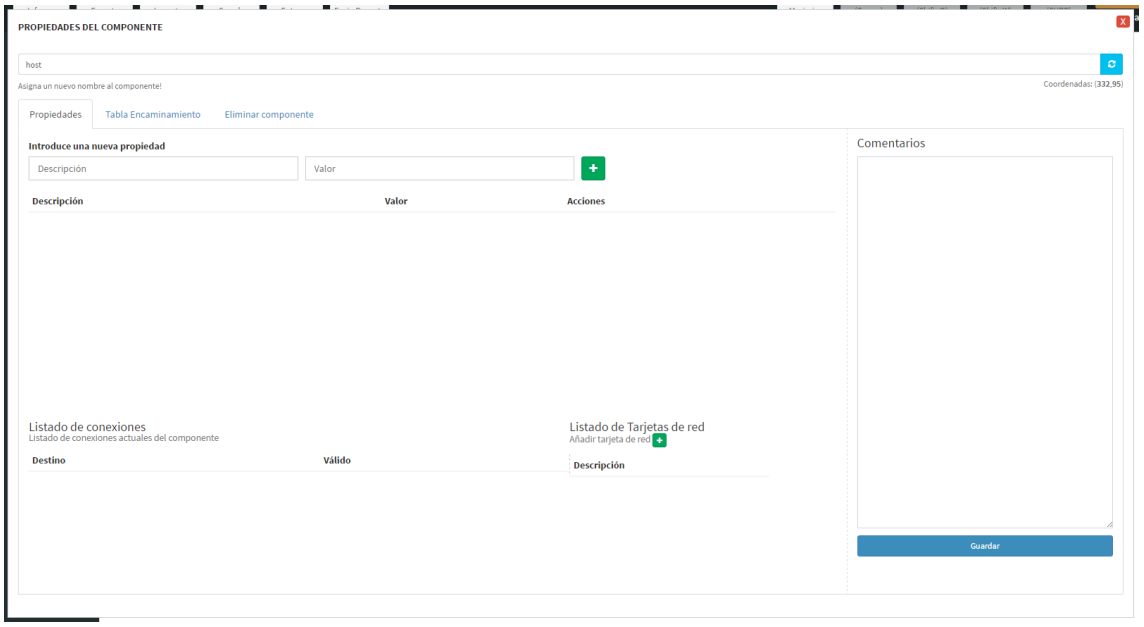

Ilustración 50 - Ventana modal de propiedades de un componente.

En la Ilustración 22 podemos ver la ventana de propiedades de un componente.

Podemos distinguir las siguientes zonas:

- En la parte superior podemos ver el nombre del componente. Este nombre es el mismo que se le puso al registrarlo en la base de datos pero puede ser modificado por un usuario por ejp en la ilustración podemos ver que pone host y al usuario le podría ser útil poner H1, H2 o cualquier otro nombre que identifique de manera inequívoca al componente dentro de su diagrama. Este nombre es que incialmente se ve en el diagrama aunque como veremos más adelante el usuario puede seleccionar qué propiedad es la que desea que se vea en el lienzo. Si desea cambiar el nombre deberá pulsar el botón azúl que hay en la parte derecha del cuadro de texto donde aparece el nombre. Debajo vemos las coordenadas del componente en el lienzo.
- Justo debajo disponemos de un sistema de pestañas en las que podemos ver:
    - Propiedades: Aquí se registrarán las propiedades, veremos la lista de conexiones (solo para consultar) y las tarjetas de red asociadas al componente.
    - Tabla de encaminamiento: Aquí podrá registrarse toda la información sobre la tabla de encaminamiento del componente. Esta información es útil cuando se van a realizar envío de paquetes entre componentes.
    - Eliminar componente: nos permite eliminar de manera individual un componente. Siempre nos pedirá confirmación para realizar el borrado. Esta misma acción sería la equivalente a pulsar la tecla SUPR tras haber seleccionado el componente.
- En la parte derecha podemos ver un cuadro de texto donde el usuario puede dejar algún comentario sobre el componente bien sea a modo de recordatorio o para dar las explicaciones solicitadas durante una prueba de examen.

AN5.1: Propiedades.

En la pestaña de propiedades el usuario podrá añadir al componente todas las propiedades que desee con el fin de configurar el componente correctamente.

Para hacerlo tan solo tendrá que rellenar la descripción y el valor y pulsar el botón verde con el símbolo del “+” que hay justo al lado de la casilla de valor.

Si la propiedad tiene rellenos tanto la descripción como el valor y siempre y cuando no exista ya una propiedad con la misma descripción, el sistema añadirá dicha propiedad al componente y estará lista para ser validada en caso de que no estemos en un modo examen.

Las diferentes respuestas que daría el sistema ante la validación de una propiedad se ilustran en las siguientes imágenes

Ilustración 51 - Intento de añadir una propiedad repetida

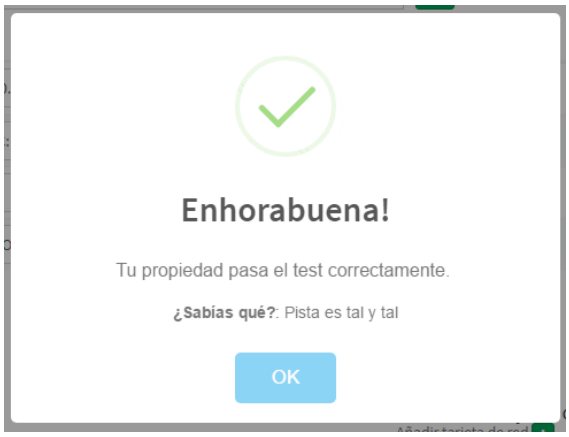

Ilustración 52 - Propiedad que pasa la validación

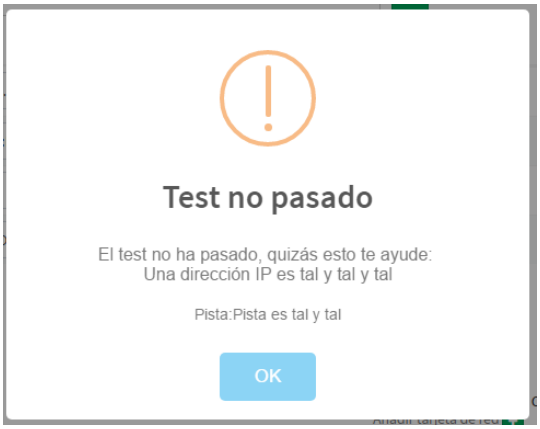

Ilustración 53 - Propiedad que no pasa la validación pero su nombre es válido

Como se puede ver en la ilustración 23 el sistema impide que introduzcamos propiedades repetidas para un mismo componente.

En la ilustración 24 podemos ver un ejemplo en el que la propiedad se ha validado correctamente y se ofrece al usuario una pista en modo de “Sabías qué?”.

En la ilustración 25 podemos ver un ejemplo en el que el nombre de la propiedade es válido pero el valor introducido no lo es. Como no estamos en modo examen el sistema proporcionará una pista para que el usuario pueda intuir cuál ha sido su error.

Toda esta información es persistente de modo que el usuario sabe en todo momento cuáles han sido los errores que ha cometido y cuáles no. Esto podemos verlo en la siguiente imagen donde se nos muestra cuáles son las propiedades que han pasado las expresiones regulares y cuáles no.

Además, sobre este listado podemos observar diferentes operaciones a realizar:

- Actualizar: El primer botón azul nos permite actualizar el valor de una propiedad.
- Validar: El botón amarillo nos permite realizar una validación sobre una propiedad.
- Borrar: El botón rojo nos permite eliminar una propiedad
- Visualizar: El último botón azul nos permite hacer que en vez del nombre se muestre el valor de esta propiedad en el lienzo. Esto nos permite que en vez de ver Host1, podamos por ejemplo, ver el valor de la IP asociada al componente.

AN5.2: Listado de conexiones y tarjetas de red

Justo debajo del listado de propiedades podremos ver el listado de conexiones del componente con el resto de componentes y las interfaces físicas de las que dispone para realizar conexiones.

Para poder realizar conexiones entre diferentes componentes debemos haberlos seleccionado en el lienzo y deben disponer de interfaces físicas disponibles para realizar las conexiones entre ellos.

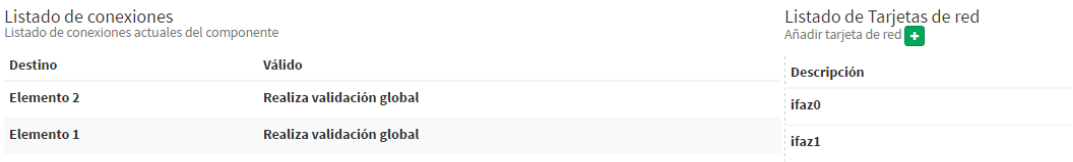

Ilustración 54 - Información sobre conexiones y tarjetas de red.

En la ilustración 26 podemos ver cómo el usuario ha realizado dos conexiones con este componente. Estas conexiones son el Elemento 2 y el Elemento 1 conectados a la ifaz0 y la ifaz1 respectivamente.

ANEXO II: Instalación

La instalación de nuestra aplicación es muy sencilla. Tan solo hay que copiar los ficheros en la ruta donde quiera alojarse la aplicación dentro del servidor. Este servidor, evidentemente, debe tener instalado PHP para poder correr y tener acceso a una base de datos MySQL.

Una vez los ficheros estén en su sitio debe ejecutarse el fichero esquemaBD.sql en la base de datos.

Por último nos quedaría configurar la conexión con la base de datos en el fichero Conexion.php que se encuentra en la carpeta request/core de nuestra aplicación.

Esta clase contiene una serie de atributos privados como son:

$host: Indica le host donde se aloja nuestra base de datos.
$username: Indica el nombre de usuario con acceso a la base de datos de nuestra aplicación.
$pwd: Indica el password asociado al usuario.
$database: Indica le nombre de la base de datos de nuestra aplicación.

Una vez realizados estos pasos nuestra aplicación debería funcionar correctamente.